Excel VBA与VSTO基础实战指南

罗刚君 著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

VISIT…

LANZAROTE Caliente.com

内 容 简 介

《Excel VBA与VSTO基础实战指南》属于学习Excel VBA的初中级教材，重点阐述了Excel VBA的基本理论、代码自动化以及开发Excel插件的思路。通读本书可以轻松应对制表工作中的疑难，同时还可以开发商业插件。

本书包括五部分内容，第一部分简述自动化操作的意义和成果展示；第二部分详细剖析VBA 的所有基础概念，包含代码的存放位置、写书方式、调用方式，认识对象、属性、方法与事件，以及理解变量、常量与数据类型并且掌握循环语句、条件语句、防错语句等知识；第三部分是VBA的高级应用，包含数组、窗体、字典、功能区设计、插件开发和撤销代码等知识；第四部分介绍通过VSTO开发Excel插件；第五部分提供365个VBA常见疑难解答。

本书每段代码都有思路分析，且对每句代码都提供了代码含义的详细注释，力求使讲解过程可以更加精准，让代码更易理解，为读者提供更优秀的阅读体验。

本书提供读者交流群，QQ群号：47700194。读者在阅读过程中有任何疑问可以加群参与讨论，同时也可以在群里下载随书案例文件和所有课后练习题的答案。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有，侵权必究。

图书在版编目（CIP）数据

Excel VBA 与 VSTO 基础实战指南 / 罗刚君著. —北京：电子工业出版社，2017.9

ISBN 978-7-121-32003-3

Ⅰ. ①E… Ⅱ. ①罗… Ⅲ. ①表处理软件②BASIC 语言—程序设计 Ⅳ. ①TP391.13②TP312.8

中国版本图书馆 CIP 数据核字(2017)第 140815 号

策划编辑：张慧敏

责任编辑：牛　勇

印　　刷：

装　　订：

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱　　邮编：100036

开　　本：787×1092　1/16　印张：33　字数：866 千字

版　　次：2017 年 9 月第 1 版

印　　次：2017 年 9 月第 1 次印刷

印　　数：3000 册　　定价：79.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：(010) 88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819，faq@phei.com.cn。

前　　言

Microsoft Excel 是制表工具中最强大的软件，但是 Excel 内置的功能无法满足相对复杂的工作需求，而且不具备自动化操作的特性。在此前提下，VBA 应运而生，它可以让复杂操作简单化，让烦琐工作自动化。

目前 Microsoft Office 已升级到 2016 版，Excel 自带的 VBA 版本为 7.1。本书以 Excel 2016 为基础编写，但是由于 Excel 2010、Excel 2013 和 Excel 2016 的 VBA 的差异微乎其微，小到可以忽略，因此读者也可以使用 Excel 2010、Excel 2013 来学习本书的知识。

本书目的

本书重点有两个，其一是普及 Excel VBA 基础知识，其二是开发 Excel 插件。

本书用了超过 50%的篇幅阐述 Excel VBA 的基本概念与语法规则，力图使读者在强化编程理论知识的基础上再学习插件开发等进阶应用。在基础章节中，重点展示了过程、参数、变量、常量、数据类型、对象、属性、方法和事件等概念，并通过第 7 章和第 8 章的数十个案例印证这些理论的价值，以及调用思路。

本书从第 13 章开始，重点讲述开发 Excel 插件的知识和步骤，并提供了诸多模板供读者调用。Excel 插件从大体上分为两类，其一是开源的加载宏文件，其二是受保护的加载项。

加载宏通常为 xla 或者 xlam 格式，直接在 VBA 的代码编译器中编写，本书不仅详细演示了开发加载宏的所有步骤和思路，还提供了让插件执行后可以撤销的方法，让用户在使用过程中不用担心覆盖重要数据，从而提升插件的品质。

加载项通常是 DLL 格式，使用 VB 或者 VB.net 开发，VB 开发的插件无法用于 64 位的 Office 软件，因此本书为读者展示了 VB.net 中的 Office 插件开发工具 VSTO 的应用。

VSTO 属于 Visual Studio 平台中的工具，本书以 Visual Studio 2015 为例，详细分析了 VSTO 与 VBA 代码的语法差异，并通过三个最具代表性的插件设计步骤演示利用 VSTO 封装代码的过程，同时也为读者提供诸多模板，让大家在实际工作中利用这些模板快速地设计出自己的插件。

本书结构

《Excel VBA 与 VSTO 基础实战指南》大体分为五部分：

第一部分包含第 1 章，主要说明 VBA 的价值，从而提升读者对 VBA 的学习热情与兴趣。

第二部分最重要，详细剖析了 VBA 的基础理论，包含第 2 章到第 9 章。此部分内容重点展示了 Excel VBA 的所有基础理论，包含代码的存放位置、输入代码的方式、调用代码的方式、如何让代码运行时畅通无阻，理解什么是过程、对象、参数、事件、属性、方法、变量、常量和数据类型，并逐一讲解了工作中最有用的条件语句、循环语句和防错语句的语法，同时提供了数十个案例来加深读者对这些基础理论的理解。

第三部分包含第 10 章到第 15 章，分别介绍了通过数组优化代码，利用字典去除重复值，以及设计功能区中的菜单的思路，并提供了大量的模板。最后讲解开发通用插件，并让插件在执行过程中可以撤销，这是本书的一大特色内容。

第四部分属于 VSTO 的应用，也就是利用 VB.net 来封装 VBA 代码，将它打包成受保护的安装程序，从而提升插件代码的安全性和专业性。

第五部分包含第 20 章，本章为读者提供 365 个 VBA 思考题目，并在赠送的案例文件中提供了答案。

本书特点

相比同类书籍，本书在内容编排上具有以下特点：

1．本书对于 Excel VBA 的基础理论有着相当详细的讲解，包含 200 多页，8 个章节。要学好编程必须基础理论掌握通透，否则编写三五年代码后仍然不能得心应手。基础理论是程序员十分重要的必备素质，而不应该只重实战轻理论。

2．目前国内 VBA 图书讲插件开发思路的书极少，而市场对插件的需求却极大，开发插件有较广阔的前景。本书不仅用较大的篇幅讲述插件开发相关的知识，而且加入了执行插件命令后可以撤销的设计思路，这在 VBA 图书市场上绝无仅有。

3．本书除 VBA 外，还提供 VSTO 知识，采用 VB.net 语法编写。

VSTO 比 VBA 更强大，能实现的功能也更多，同时还更安全。本书详细罗列了 VBA 与 VSTO 在代码上的差异，然后演示修改 VBA 代码，使其符合 VB.net 语法规则的基本思路，从而让读者快速学会利用 VSTO 开发 Excel 插件。掌握本书第 16 章到第 19 章的知识，仅需半个月即可学会 VSTO，不需要像学习 C#那样，耗费半年甚至一两年时间。

4．本书在每章末尾会提供 5 个思考题，最后一章再追加 365 个思考题，一共 460 个。期望读者通过这些题目扩展知识面，同时能加深对书中理论知识的印象。

5．本书提供读者交流群，读者购书后可以加群下载案例文件，同时可以在群里与作者交流，加快学习进度。

案例文件

本书不提供光盘，请加入交流群下载，也可以到电子工业出版的博文视点官方网站下载，网址如下：

http://www.broadview.com.cn/32003

本书作者

本书由罗刚君编写，罗刚君是多个大型论坛的版主，有着丰富的 VBA 程序设计经验。

作者近 10 年来已出版 15 部关于 Excel 的图书（含本书），分别是《Excel 2007 VBA 范例大全》、《Excel 2007 技法与行业应用实例精讲》、《Excel 2007 VBA 开发技术大全》、《Excel 2007 函数案例速查宝典》、《Excel VBA 程序开发自学宝典》、《Excel 2010 VBA 编程与实践》、《Excel 2010

函数与图表速查手册》、《Excel VBA 程序开发自学宝典（第 2 版）》、《Excel 函数、图表与透视表从入门到精通》、《来吧！带你玩转 Excel VBA》、《Excel VBA 程序开发自学宝典（第 3 版）》、《Excel 2013 函数案例自学宝典（实战版）》、《Excel 2013 VBA 编程与实践》、《Excel 2016 实用技巧自学宝典》、《Excel VBA 与 VSTO 基础实战指南》。

读者在阅读本书过程中可以在群里随时与作者沟通，或者反馈阅读过程中遇到的问题，同时也可以向作者提出有关 VBA 或者 VSTO 的建议，作者会利用业余时间及时回复。

请扫码加入交流群

作者：罗刚君

2017 年 8 月 30 日

读者服务

轻松注册成为博文视点社区用户（www.broadview.com.cn），扫码直达本书页面。

- **下载资源**：本书如提供示例代码及资源文件，均可在 下载资源 处下载。
- **提交勘误**：您对书中内容的修改意见可在 提交勘误 处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- **交流互动**：在页面下方 读者评论 处留下您的疑问或观点，与我们和其他读者一同学习交流。

页面入口：http://www.broadview.com.cn/32003

目　　录

第 1 章　自动化操作 1

1.1　自动化操作的价值 1
　1.1.1　自动化操作的必要性 1
　1.1.2　实现自动化操作的两个途径 3
1.2　利用宏简化日常工作 3
　1.2.1　三分钟学会录制宏 4
　1.2.2　执行宏的方法 7
　1.2.3　两种方法读懂宏代码 7
　1.2.4　宏的优缺点分析 10
　1.2.5　如何发挥宏的长处 11
1.3　使用 VBA 强化 Excel 功能 12
　1.3.1　追根溯源：什么是 VBA 12
　1.3.2　知己知彼：解析 VBA 的优缺点 13
　1.3.3　窥斑见豹：从一个案例初识 VBA 13
1.4　Excel VBA 的发展前景 15
　1.4.1　简化工作 15
　1.4.2　开拓专业 16
1.5　课后思考 16

第 2 章　代码应用基础 17

2.1　区分 VBE 代码窗口 17
　2.1.1　认识 VBE 窗口 17
　2.1.2　最常用的代码存放区：标准模块 19
　2.1.3　工作簿事件代码窗口：ThisWorkbook 23
　2.1.4　工作表事件代码窗口：Sheet1 23
　2.1.5　窗体代码窗口：UserForm1 24
　2.1.6　创建隐藏对象的代码窗口：类模块 24
2.2　录入代码 25
　2.2.1　代码的存放位置 25
　2.2.2　写入代码的方式 25
　2.2.3　提升代码的可读性 27
　2.2.4　调用快速信息 31

2.3 四种代码执行方式 32
2.3.1 调用快捷键 32
2.3.2 单击按钮执行 33
2.3.3 自动执行 34
2.3.4 在公式中调用 35
2.4 保存代码 36
2.4.1 修改文件的保存格式 36
2.4.2 一劳永逸 37
2.5 让代码畅通无阻 37
2.5.1 调整宏的安全等级 38
2.5.2 添加受信任位置 38
2.5.3 将代码封装为加载项 39
2.6 反复调用相同代码 39
2.6.1 使用个人宏工作簿 39
2.6.2 加载宏 40
2.6.3 加载项 40
2.7 课后思考 40

第 3 章 从概念开始认识 VBA 42

3.1 认识过程 42
3.1.1 过程的分类 42
3.1.2 Sub 过程的基本语法 42
3.1.3 Sub 过程的命名要求 44
3.1.4 Sub 过程的调用方法与访问限制 45
3.1.5 过程的执行顺序 46
3.1.6 过程的递归 48
3.2 关于参数 49
3.2.1 参数的存在价值 49
3.2.2 过程名称中的参数 49
3.2.3 参数的赋值方式 50
3.2.4 可选参数与必选参数 52
3.2.5 代码中的参数 53
3.3 理解对象 54
3.3.1 什么是对象 54
3.3.2 对象的引用层次 55
3.4 对象的属性与方法 56
3.4.1 认识属性与方法 56
3.4.2 自动调用属性与方法 57
3.4.3 怎样才算完整的 VBA 语句 58
3.5 对象的事件 59

3.5.1 什么是事件 ... 59
3.5.2 事件的存在价值 ... 60
3.5.3 事件的分类与代码录入方式 ... 60
3.5.4 事件的参数 ... 62
3.6 课后思考 ... 63

第 4 章 对象及其层次结构 ... 65

4.1 查看所有对象 ... 65
4.1.1 从对象浏览器查看对象 ... 65
4.1.2 从帮助中调用对象的详细信息 ... 65
4.2 对象的层次与引用方式 ... 67
4.2.1 对象的层次 ... 67
4.2.2 使用对象名称引用对象 ... 67
4.2.3 使用复数形式表示对象集合 ... 68
4.2.4 使用序号参数引用集合中的子对象 ... 69
4.2.5 引用子对象 ... 70
4.2.6 引用活动对象 ... 70
4.2.7 引用父对象 ... 71
4.2.8 利用 With 语句引用重复出现的对象 ... 72
4.3 Range 对象 ... 74
4.3.1 Range("A1")引用方式 ... 74
4.3.2 Cells(1,1)引用方式 ... 76
4.3.3 [A1]引用方式 ... 77
4.3.4 活动单元格：ActiveCell ... 78
4.3.5 屏幕坐标下的单元格：RangeFromPoint ... 79
4.3.6 选区：Selection、RangeSelection ... 80
4.3.7 已用区域：UsedRange ... 81
4.3.8 当前区域：CurrentRegion ... 83
4.3.9 按条件引用区域：SpecialCells ... 83
4.3.10 模拟 End+方向键产生的单元格：End ... 86
4.3.11 按偏移量重置区域引用：Offset ... 88
4.3.12 按宽度与高度重置区域：Resize ... 90
4.3.13 引用多区域的合集：Union ... 91
4.3.14 引用多区域的交集：Intersect ... 92
4.4 图形对象 ... 95
4.4.1 Shapes 对象与子对象 ... 95
4.4.2 图形对象的名称 ... 95
4.4.3 DrawingObjects ... 96
4.4.4 图形对象的类别子集 ... 97
4.5 表对象 ... 98

4.5.1 表的合集与子对象........98
4.5.2 表对象的分类........98
4.5.3 活动表........99
4.5.4 隐藏工作表的特性........100
4.5.5 引用名字为数值的工作表的技巧........100
4.6 工作簿对象........101
4.6.1 工作簿合集与子对象........101
4.6.2 活动工作簿........101
4.6.3 关于后缀名........102
4.6.4 关于工作簿格式........103
4.7 Excel 应用程序对象........103
4.7.1 Excel 的顶层对象：Application........104
4.7.2 调用子对象时可以省略 Application 吗........104
4.7.3 不同版本的 Excel 之间的差异........104
4.8 课后思考........105

第 5 章 揭密数据类型与变量、常量........106

5.1 数据类型........106
5.1.1 区分数据类型的必要性........106
5.1.2 数据类型的分类........107
5.1.3 转换数据类型........110
5.2 定义变量........112
5.2.1 变量的用途........112
5.2.2 定义变量的方法........113
5.2.3 变量的命名规则........114
5.2.4 变量的作用域........115
5.2.5 变量的生命周期........117
5.2.6 静态变量与动态变量的区别........118
5.2.7 声明对象变量........119
5.2.8 对象变量的初始化与释放........121
5.3 定义常量........122
5.3.1 常量的用途........122
5.3.2 常量的定义方式........122
5.3.3 变量与常量的异同分析........123
5.4 课后思考........123

第 6 章 条件语句与循环语句........125

6.1 If 语句解析........125
6.1.1 条件语句的重要性........125

6.1.2 If...Then...Else 的单行模式 126
6.1.3 And、Or 和 Not 在条件语句中的作用 127
6.1.4 案例解析：指定工作簿的最后开启日期 128
6.1.5 If...Then...Else 的块形式 129
6.1.6 块形式的应用案例：创建日期批注 130
6.1.7 嵌套使用 If 语句 132
6.1.8 If 语句的常见错误与防错之法 138
6.2 Select Case 语句解析 140
6.2.1 Select Case 语句的价值 140
6.2.2 Select Case 基本语法 140
6.2.3 多条件应用案例 142
6.3 IIf 函数 145
6.3.1 IIf 函数语法解析 146
6.3.2 IIf 函数案例应用：判断 Excel 的版本号 146
6.3.3 IIf 函数的优缺点 147
6.4 For Next 语句解析 147
6.4.1 循环语句的作用 147
6.4.2 For Next 语句基本语法 148
6.4.3 步长值对循环结果的影响 149
6.4.4 For Next 循环语句应用案例 149
6.5 For Each…Next 语句解析 153
6.5.1 遍历对象集合 153
6.5.2 For Each…Next 语句基本语法 153
6.5.3 For Each…Next 语句应用案例：定位大于某值的单元格 154
6.6 Do Loop 语句解析 156
6.6.1 Do Loop 语法分析 156
6.6.2 Do Loop 语法一应用 158
6.6.3 Do Loop 语法二应用 161
6.6.4 Do Loop 语法三应用 162
6.6.5 Do Loop 语法四应用 163
6.6.6 总结三种循环语句的优缺点 166
6.7 课后思考 166

第 7 章 四类常见对象的应用案例 168

7.1 单元格对象 168
7.1.1 选择单元格 168
7.1.2 筛选与复制区域的值 169
7.1.3 多区域复制 171
7.1.4 选择性粘贴数据 172
7.1.5 重置已用数据区域 175

7.1.6 查找所有成绩为 100 的单元格 177
7.1.7 将表示平方米和立方米后面的 2 和 3 设为上标 178
7.1.8 合并相邻且相同的单元格 180
7.1.9 按行合并且保留所有数据 182
7.1.10 隔行插入行 183
7.1.11 标示选区中的重复值 184
7.2 图形对象......186
7.2.1 批量导入图片与图片名称 186
7.2.2 统一表中所有图片大小及对齐图片 189
7.2.3 插入图片到选区中 190
7.2.4 插入带图片背景的批注 192
7.3 工作表对象......194
7.3.1 显示所有隐藏的工作表 194
7.3.2 创建以本月每日日期命名的工作表 195
7.3.3 保护所有公式 196
7.3.4 批量重命名表 198
7.3.5 查找所有工作表中有循环引用的单元格......199
7.3.6 对职工表按学历排序......200
7.3.7 创建工作表目录......202
7.4 工作簿对象......204
7.4.1 打开带密码且带有自动宏的工作簿 204
7.4.2 另存工作簿且以今天的日期命名 204
7.4.3 将外部链接转换成值......205
7.4.4 关闭工作簿且不保存修改内容......206
7.4.5 定时保存且备份工作簿 207
7.4.6 重命名活动工作簿......208
7.5 课后思考......211

第 8 章 深入剖析 VBA 的各种事件 212

8.1 事件的级别与顺序 212
8.1.1 事件的级别与代码保存位置......212
8.1.2 事件的执行方式......214
8.1.3 事件的执行顺序......215
8.2 禁用与启用事件 215
8.2.1 临时关闭事件 215
8.2.2 防止事件的连锁反应 216
8.3 工作表事件详解 217
8.3.1 工作表事件列表......217
8.3.2 Change 事件的特例 218
8.3.3 事件案例：激活工作表时验证访问权限......219

8.3.4 事件案例：自动标示当前行的背景 220
8.3.5 事件案例：双击单元格时选中所有相同值 222
8.3.6 事件案例：在特定区域右击单元格时产生工作表目录 223
8.3.7 事件案例：输入表达式时在右列自动返回计算结果 224
8.3.8 事件案例：单击目录时可打开隐藏的工作表 225
8.3.9 事件案例：实时保护已录入数据的单元格 226
8.3.10 事件案例：在状态栏显示当前科目的不及格人数 227
8.3.11 事件案例：通过数据有效性的下拉列表调用对应的图片 228
8.4 工作簿事件详解 229
8.4.1 工作簿事件列表 230
8.4.2 事件案例：记录工作簿打开次数 231
8.4.3 事件案例：显示活动工作表中的产量达标率 232
8.4.4 事件案例：打印数据前检查资料是否填写完整 234
8.4.5 事件案例：保存工作簿时更新工作表目录 235
8.4.6 事件案例：新建工作表时调用模板格式 236
8.4.7 事件案例：禁止修改总表名称 237
8.4.8 事件案例：新建图表时自动设置为阴影、圆角 238
8.5 应用程序级事件详解 239
8.5.1 应用程序与类 239
8.5.2 事件案例：打开任意工作簿时创建工作表目录 239
8.5.3 事件案例：新建工作簿时自动保存 241
8.6 按时间执行代码 242
8.6.1 OnKey 方法的语法分析 243
8.6.2 创建计划任务 243
8.7 课后思考 244

第 9 章 处理代码错误 245

9.1 代码错误类型分析 245
9.1.1 版本问题 245
9.1.2 参数赋值不当 245
9.1.3 变量定义不准确 247
9.1.4 对象不存在 247
9.2 错误处理语句 248
9.2.1 详解 Err 对象 249
9.2.2 Error 函数详解 250
9.2.3 On Error Resume Next 语句 250
9.2.4 On Error GoTo Line 语句 252
9.2.5 On Error GoTo 0 语句 254
9.2.6 Gosub...Return 语句 254
9.3 案例应用 258

9.3.1 处理错误的常规思路 258
9.3.2 案例应用：按条件定位单元格 259
9.3.3 案例应用：根据选区的文件名批量导入图片 262
9.3.4 案例应用：一键屏蔽错误值 264
9.4 课后思考 265

第 10 章 使用数组提升程序效率 267

10.1 基本概念 267
10.1.1 何为数组 267
10.1.2 数组的特点 267
10.1.3 一维数组 268
10.1.4 二维数组 270
10.1.5 数组的参数 271
10.1.6 声明数组变量 272
10.1.7 动态数组与静态数组的区别 275
10.1.8 释放动态数组的存储空间 280
10.2 数组函数 281
10.2.1 用函数创建数组 281
10.2.2 获取数组元素 282
10.2.3 判断变量是否为数组 283
10.2.4 转置数组 283
10.2.5 获取数组的上标与下标 285
10.2.6 转换文本与数组 286
10.2.7 筛选数组 288
10.3 案例分析 289
10.3.1 将指定区域的单词统一为首字母大写 289
10.3.2 罗列不及格学生姓名、科目和成绩 290
10.3.3 将字符串合并到区域 292
10.3.4 将职员表按学历拆分成多个工作表 294
10.3.5 将选区的数据在文本与数值间互换 297
10.3.6 获取两列数据的相同项 298
10.3.7 罗列至少三科不及格的学生姓名 300
10.4 课后思考 302

第 11 章 集合与字典的应用 303

11.1 Collection:集合 303
11.1.1 集合的特性 303
11.1.2 集合的语法 304
11.1.3 使用集合获取区域中的不重复值 307

11.1.4 罗列 B 列重复出现的身份证号码 308
11.2 Dictionary:字典 310
11.2.1 字典对象的前期绑定和后期绑定 310
11.2.2 字典的特点 312
11.2.3 字典的属性与方法 312
11.2.4 获取选区中的唯一值 317
11.2.5 对采购表分类求和 318
11.2.6 对采购表分类计数 319
11.2.7 对产量表按组别和产品分类统计 320
11.3 课后思考 321

第 12 章 设计程序窗体 323

12.1 窗体与控件简介 323
12.1.1 窗体的功能 323
12.1.2 创建与运行 UserForm 对象 325
12.1.3 使用工具箱 326
12.1.4 标签控件 329
12.1.5 文本框控件 329
12.1.6 命令控钮 329
12.1.7 复合框 330
12.1.8 列表框 330
12.1.9 复选框 331
12.1.10 选项按钮 331
12.1.11 框架 331
12.1.12 切换按钮 332
12.1.13 多页控件 333
12.1.14 滚动条 334
12.1.15 图像控件 334
12.1.16 Flash 控件 334
12.2 设置属性 335
12.2.1 属性窗口的用途 335
12.2.2 设置属性的两种方式 336
12.2.3 文本框属性 338
12.2.4 命令按钮属性 341
12.2.5 复选框属性 343
12.2.6 列表框属性 345
12.2.7 复合框属性 350
12.2.8 图像控件属性 353
12.2.9 Flash 控件属性 354
12.2.10 批量设置控件的属性 354

12.3 窗体与控件的事件......355
12.3.1 UserForm 对象的事件......355
12.3.2 控件的事件......356
12.4 窗体应用实战......366
12.4.1 开发多工作表查询窗体......366
12.4.2 开发多工作表快速录入面板......369
12.4.3 以指定名称批量新建或复制工作表......371
12.5 课后思考......375

第 13 章 定义 Ribbon 功能区选项卡......377

13.1 功能区选项卡开发基础......377
13.1.1 Ribbon 的特点......377
13.1.2 功能区的组件图示......377
13.1.3 手动定制功能区......378
13.1.4 认识 Ribbon 代码编辑器......378
13.1.5 获取内置按钮图标......379
13.2 Ribbon 定制之语法分析......380
13.2.1 功能区代码的结构......380
13.2.2 显示与隐藏功能区：ribbon......382
13.2.3 创建新选项卡：tab......382
13.2.4 创建新组：group......384
13.2.5 创建对话框启动器：dialogBoxLauncher......385
13.2.6 在组中添加命令按钮：button......387
13.2.7 创建切换按钮：toggleButton......389
13.2.8 创建弹出式菜单：menu......390
13.2.9 创建下拉列表：dropDown......391
13.2.10 创建编辑框：editBox......393
13.2.11 锁定或隐藏内置功能......394
13.3 使用回调函数强化功能区......395
13.3.1 为什么需要使用回调函数......395
13.3.2 回调函数详解......395
13.3.3 创建 1 到 3 号才能使用的按钮......397
13.3.4 创建按下与弹起时自动切换图标的按钮......398
13.3.5 在功能区中快速查找......400
13.3.6 在组的标签处显示日期及问候语......403
13.3.7 调用大图片创建下拉菜单......404
13.4 使用模板......408
13.4.1 模板的重要性......408
13.4.2 模板的使用方法......408
13.4.3 制作两个模板......408

13.5 课后思考 412

第 14 章 开发通用插件 413

14.1 插件的分类 413
14.1.1 什么是插件 413
14.1.2 插件的分类方式 413
14.1.3 开发插件和编写普通代码的分别 414
14.2 漫谈加载宏 414
14.2.1 加载宏工作簿的特点 414
14.2.2 加载宏管理器 415
14.2.3 加载宏的使用方法 416
14.2.4 加载宏的便利性 417
14.3 制作工作表批量重命名插件 417
14.3.1 开发通用插件的基本步骤 417
14.3.2 罗列插件需求 417
14.3.3 设计插件窗体 418
14.3.4 编写代码 419
14.3.5 创建菜单与设置快捷键 422
14.3.6 安装并测试功能 422
14.4 课后思考 425

第 15 章 让 VBA 代码也能撤销 426

15.1 突破撤销限制 426
15.1.1 VBA 命令的撤销限制 426
15.1.2 设计可以撤销的 Sub 过程的思路与步骤 427
15.2 设计可撤销的插件 428
15.2.1 编写插件 428
15.2.2 为插件添加撤销功能 431
15.3 课后思考 436

第 16 章 使用 VSTO 设计插件的基本步骤 437

16.1 安装 Visual Studio 2015 437
16.1.1 VSTO 对于 Excel 用户的意义 437
16.1.2 Visual Studio 版本介绍 438
16.1.3 安装 Visual Studio 2015 438
16.2 Excel 插件开发流程 440
16.2.1 创建项目 440
16.2.2 设计功能区 441
16.2.3 写入 Sub 过程 442

16.2.4 生成 DLL 插件 443
16.3 将插件打包成安装程序 444
16.3.1 Inno Setup 软件介绍 444
16.3.2 打包插件安装程序 444
16.3.3 安装插件 447
16.4 课后思考 449

第 17 章 VSTO 与 VBA 的差异 450

17.1 变量、常量与数据类型 450
17.1.1 数据类型 450
17.1.2 变量 451
17.1.3 常量 452
17.2 函数 452
17.2.1 调用方式不同 453
17.2.2 函数差异 453
17.3 数组 454
17.3.1 原本功能的差异 454
17.3.2 新增功能 455
17.4 窗体 455
17.4.1 名称变化 455
17.4.2 调用方式变化 456
17.4.3 功能变化 458
17.5 字典与正则表达式 459
17.5.1 字典 459
17.5.2 正则表达式 459
17.6 菜单与功能区 460
17.6.1 工作表菜单 460
17.6.2 功能区菜单 460
17.7 管理文件与目录 461
17.7.1 管理文件 462
17.7.2 管理目录 462
17.8 杂项 463
17.9 课后思考 465

第 18 章 将 VBA 插件升级为 VSTO 插件 466

18.1 设计插件框架 466
18.1.1 VBA 插件介绍 466
18.1.2 设计插件框架 467
18.2 升级 Sub 过程“创建工资条” 468

18.2.1 准备工作 468
18.2.2 修改代码 469
18.3 升级窗体“文件批量命名” 470
18.3.1 准备工作 471
18.3.2 修改代码 474
18.4 升级事件过程“零值控制器” 477
18.4.1 编写功能区回调过程 477
18.4.2 编写应用程序级事件过程 477
18.5 打包安装程序 478
18.5.1 制作安装程序 478
18.5.2 安装测试 479
18.6 课后思考 481

第 19 章 VSTO 的更多高级应用 482

19.1 添加窗体状态栏 482
19.1.1 设计 482
19.1.2 测试 483
19.2 创建任务栏图标 484
19.2.1 设计 484
19.2.2 测试 486
19.3 自动发邮件 486
19.3.1 设计 486
19.3.2 测试 488
19.4 全自动合并数据 488
19.4.1 设计 489
19.4.2 测试 493
19.5 设计任务窗格 494
19.5.1 设计 494
19.5.2 测试 496
19.6 课后思考 498

第 20 章 365 个 VBA 常见问题答疑 499

第 1 章　自动化操作

面对日益复杂的工作需求，自动化操作有着越来越重大的意义。

像在 DOS 系统中提供 BAT 格式的批处理文件一样，Excel 也提供宏与 VBA 为 Excel 实现制表自动化操作，从而在解放双手的同时也减少了重复性操作中潜藏的失误。

本章主要介绍如何通过宏与 VBA 实现简单的自动化操作，而对于代码的基本规则、语法、编写方式和调用思路则在后面的章节详细阐述。

本章要点

◆ 自动化操作的价值

◆ 利用宏简化日常操作

◆ 使用 VBA 强化 Excel 功能

◆ Excel VBA 的发展前景

1.1　自动化操作的价值

价值无疑是动力的源泉，而且价值和动力在整体的势态中往往成正比。学习 VBA 之前有必要了解它对工作能带来多少价值，是否值得投入时间和精力去学习。

1.1.1　自动化操作的必要性

自动化操作蕴含以下两项内容。

1．多个操作步骤一次性完成

“选择 A1:B11 区域，然后创建圆环图表，并对图表区设置阴影、圆角”。完成此要求需要单击鼠标 10 次以上，如果使用宏或者 VBA 则能一键完成，大幅提升工作效率。

打开随书案例文件“1-1 自动生成图表且设置阴影圆角.xlsm”，单击“一键生成图表”按钮可以实现以上所有步骤的同等效果，产生的图表如图 1-1 所示。

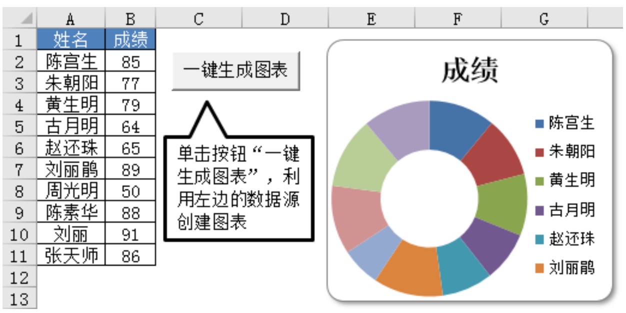


图 1-1　一键创建图表且设置其格式

在本例的文件中使用了 VBA 程序，工作表中的按钮“一键生成图表”已关联到该程序，在单击按钮时可以调用程序代码，程序依照设定的流程执行代码中的每个操作。虽然代码本身包含多个步骤，但对于用户而言仅需要单击一次鼠标，这就是自动化操作的价值所在。

注意：20 世纪 90 年代曾经爆发宏病毒，为了确保文件安全，微软在升级 Office 时将 Excel 设置为默认禁止宏运行，每次打开带有宏代码的工作簿时都会禁止运行宏，且同时产生安全警告“宏已被禁用”，只有在单击“启用内容”按钮后才可以使用宏。

在学习 Excel VBA 期间，为了使用方便，应将“宏设置”选项修改为“启用所有宏”。方法是打开“Excel 选项”对话框，并进入“信任中心”，在“宏设置”中将原本“禁用所有宏，并发出通知”修改为“启用所有宏”，操作步骤如图 1-2 所示。

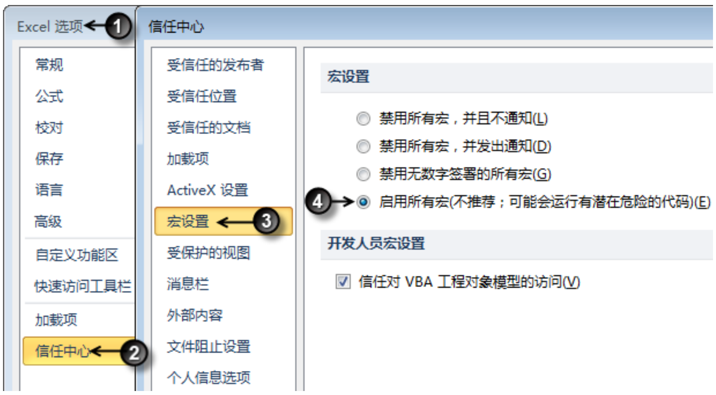


图 1-2　启用所有宏

2．重复性工作只操作一次

如果某些操作需要每天或者每月重复执行，那么最理想的方式是在第一次操作时就将所有工作设置完善，以后通过快捷键或者单击按钮自动执行所需重复的所有步骤。

例如：要求每月末打开各部门上传到服务器的共享文件夹中的 100 个工作簿，将它们的所有数据合并到一个工作簿中。如果逐个打开工作簿，然后逐个复制工作簿中的所有工作表，那么此操作既烦琐又低效，如果改用 VBA 代码自动化操作则三两秒钟即可完成。

再如将图 1-3 中的工资表转换成工资条形式，如果手动插入空行并复制标题从而生成工资条，那么表中有 1000 人则需要反复操作 1000 次，不仅工序繁多，而且容易出错。



	A	B	C	D	E	F	G	H
1	姓名	工号	职务	基本工资	加班费	职务津贴	扣伙食费	实发工资
2	计尚云	052	员工	1200	1200	400	200	2600
3	赵国	001	员工	1200	1200	400	200	2600
4	罗至贵	093	班长	1800	1400	450	200	3450
5	徐大鹏	090	员工	1200	1420	400	200	2820
6	朱聪	012	厂长	2400	500	800	350	3350

	A	B	C	D	E	F	G	H
1	姓名	工号	职务	基本工资	加班费	职务津贴	扣伙食费	实发工资
2	计尚云	052	员工	1200	1200	400	200	2600
3								
4	姓名	工号	职务	基本工资	加班费	职务津贴	扣伙食费	实发工资
5	赵国	001	员工	1200	1200	400	200	2600
6								
7	姓名	工号	职务	基本工资	加班费	职务津贴	扣伙食费	实发工资
8	罗至贵	093	班长	1800	1400	450	200	3450
9								
10	姓名	工号	职务	基本工资	加班费	职务津贴	扣伙食费	实发工资
11	徐大鹏	090	员工	1200	1420	400	200	2820

图 1-3　工资表与工资条对比

如果采用 VBA，那么上述功能可以自动化执行。只要单击相应按钮，以往可能需要半小时或者半天的工作仅在 2 秒钟内即可完成，而且多次调用 VBA 命令，能确保操作结果格式一致，这正是 VBA 的魅力所在。

本例案例文件请参考：..\ 第 1 章\ 1-2 一键生成工资条.xlsm

自动化操作对于制表者而言不仅仅在于速度更快，能节约操作时间，它同时还能避免失误。在工作中操作步骤越多，在重复同样动作时越容易出现操作失误，而一键完成操作则可以避免这种失误。此外它还能统一多个运算结果的格式，例如用宏对数据创建图表，不管操作几十次或者几万次，生成的图表都是一模一样的，包括大小、位置、颜色、阴影效果等，而每天手动操作一次，连续 10 天所创建的 10 个图表则可能产生多种效果，甚至可能忘记了其中某个步骤，从而无法满足需求。

基于以上两点，自动化操作对于工作量大的办公文员来说意义重大，即使在工作量不大的情况下也同样有此需求，因为它能确保多次操作的一致性，避免失误。只要在设计代码时经过多次测试，在完善代码后再将其投入使用，它会给用户带来无尽的便利。

1.1.2 实现自动化操作的两个途径

在 DOS 时代，让 DOS 命令自动化批量执行的方法是将命令写在 BAT 格式的批处理文件中，而 Excel 则允许使用宏或者 VBA 来实现自动化操作。

Excel 的宏是微软早期提供的一种用于强化 Excel 的工具，它包括宏函数和宏过程。宏函数一般在宏表中使用，或者将其定义为名称在普通工作表中使用；宏过程则通过录制宏，然后使用【Alt+F8】组合键调用宏的两个步骤发挥其功能。

宏的优点是不需要学习任何编程的理论，也不需要任何基础就可以使用宏代码完成重复性工作；宏的缺点是，相对于 VBA 来说，它的通用性不够好，灵活性不足。正因如此，微软在 1993 年发布 Excel 5.0 时开始大力推广 VBA 来代替宏。

VBA 是宏语言的升级版，操作上更复杂，使用者既需要懂得编程的理论，又得具备修改代码的功底，否则无法驾驭 VBA，无法让代码发挥其潜能。

形象地说，宏代码通过录制产生，通过按下【Alt+F8】组合键或者单击相应按钮调用，它算是 VBA 的初级阶段。当仅限于录制与调用时，我们通常称之为使用宏，而懂得修改、调试代码时，我们则称之为使用 VBA。其实使用宏与 VBA 都是利用代码来工作，并不把它们做严格区分。

当使用宏实现操作自动化时，有一些局限性，因为宏总是通过录制和【Alt+F8】组合键调用的方式来使用，而 Excel 只能录制部分操作，这直接导致某些操作不能转化为批量执行；VBA 可以将 Excel 的任何操作转换为代码，所以它不存在功能上的局限性，但是学习 VBA 较之学习录制宏与调用宏需要花费更多的精力。所以两者皆优劣并存，读者可以按需选择。在接下来的两个小节中，将会分别介绍如何通过宏和 VBA 实现自动化操作，从而全方位提升工作效率。

1.2 利用宏简化日常工作

录制宏是指利用宏记录器将当前所有操作记录下来，以代码形式描述所进行的操作。在回放该代码时可以重新执行所记录的所有操作，实现与预期一致的效果。

录制宏有诸多技巧，不过鉴于宏自身的局限性，本节仅对录制宏与调用宏做简要介绍，本书将 VBA 作为重点。另外本节仅阐述宏的使用方法，对于宏代码并不深入解析，在后面的章节会有相关的理论教学。

1.2.1　三分钟学会录制宏

录制宏表示记录当前操作，有“相对引用”和“绝对引用”之分，需要根据需求选择采用何种方式录制。

在以“绝对引用”的方式录制宏时将如实记录所操作的单元格的绝对地址，例如“B5”、“$F$10”在 VBA 中都属于绝对地址。当调用该宏时，总是如实地执行相同操作，不受活动单元格地址的影响。而在以“相对引用”的方式录制宏时，总是以活动单元格为参照原点来记录操作对象地址，所以当后期调用该宏时会以调用宏时的活动单元格作为参照原点。基于此，调用“相对引用”方式记录的宏和“绝对引用”方式记录的宏有所不同，前者不需要任何规则，总是按既定流程执行；后者则受活动单元格影响，在不同单元格调用宏将得到不同结果。比较之下，采用“相对引用”方式录制的宏更灵活、更强大，不过在调用时需要格外小心。

注意：“录制宏”菜单在“开发工具”选项卡中，对于普通用户而言开发工具并不常用，因此微软未将它显示在功能区中，需要进入“Excel”选项对话框手动调出该选项卡。具体操作如图 1-4 所示。

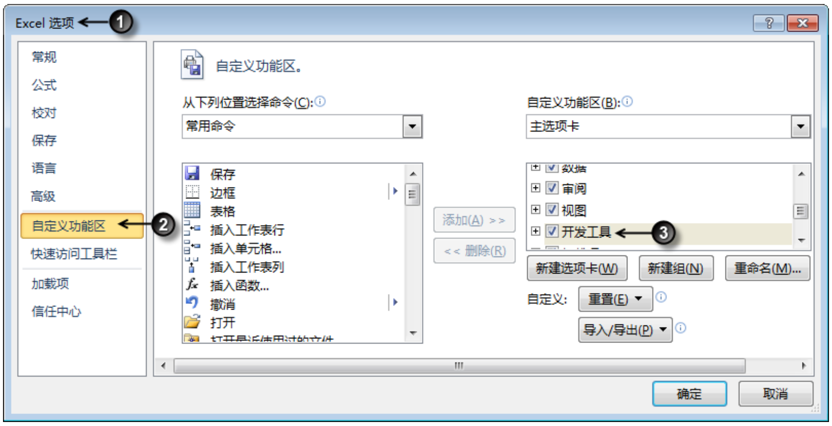


图 1–4　调出“开发工具”选项卡

下面以“相对引用”和“绝对引用”两种方式录制宏，操作步骤包括：选择 A1 单元格、录入字符串“宏与 VBA”，然后将 A1 单元格设置为加粗、下划线和 20 号字体，最后让 A 列自动适应列宽。

1. 绝对引用

新建空白工作簿并选择 B1 单元格，然后录制宏。

采用“绝对引用”方式录制宏的步骤如下。

（1）选择菜单“开发工具”→“录制宏”，打开“录制新宏”对话框，将对话框中的宏名保持默认的“宏 1”，将快捷键设置为【Ctrl+q】，并对宏加以说明，详情如图 1-5 所示。

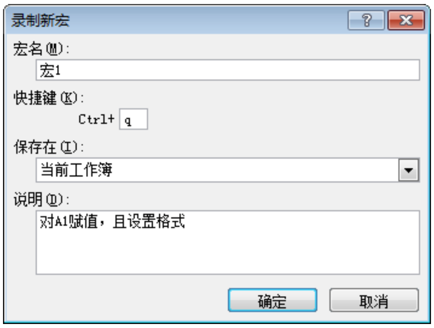


图 1-5　指定宏的快捷键和名称

（2）选择 A1 单元格，并录入文本“宏与 VBA”，然后按回车键确认录入；

（3）选择 A1 单元格，然后分别选择菜单“开始”中“字体”组对应的菜单，将 A1 单元格设置为加粗、下划线，以及 20 号字体；

（4）双击 A 列与 B 列中的分界线使 A 列自动调整列宽；

（5）选择菜单“开发工具”→“停止录制”。

以上步骤已经完成录制宏，接下来可以随时通过快捷键【Ctrl+q】执行此宏。

按【Alt+F11】组合键进入保存宏代码的界面，双击左侧工程资源管理器中的模块 1 可以看到如下宏代码（如图 1-6 所示）：

```
Sub 宏1()
' 宏1 宏
' 对A1赋值并设置格式
' 快捷键: Ctrl+q
    Range("A1").Select
    Application.FormulaBarHeight = 1
    ActiveCell.FormulaR1C1 = "宏与 VBA"
    Range("A1").Select
    Selection.Font.Bold = True
    Selection.Font.Underline = xlUnderlineStyleSingle
    With Selection.Font
        .Name = "宋体"
        .Size = 20
        .Strikethrough = False
        .Superscript = False
        .Subscript = False
        .OutlineFont = False
        .Shadow = False
        .Underline = xlUnderlineStyleSingle
        .ThemeColor = xlThemeColorLight1
        .TintAndShade = 0
        .ThemeFont = xlThemeFontMinor
    End With
    Columns("A:A").EntireColumn.AutoFit
End Sub
```

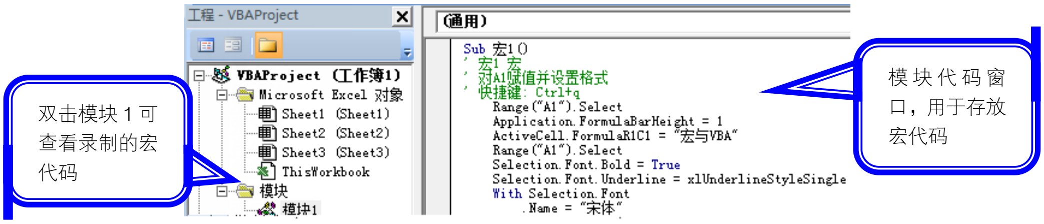


图 1-6　保存宏代码的位置

2．相对引用

删除刚才在 A 列录入的数据，然后以相对引用方式再次录制宏，本操作的目的是观察两种录制方式所产生的宏代码有何差异，以及比较两种宏代码的执行方式的异同。

仍然先选择 B1 单元格，然后再录制宏。

（1）选择菜单“开发工具”→“使用相对引用”，从而切换到相对引用状态；

（2）选择菜单“开发工具”→“录制宏”，在弹出的对话框中保持默认的宏名“宏 2”，将快捷键设置为“Ctrl+Shift+q”，然后对宏添加说明“以相对引用录制宏”；

（3）选择 A1 单元格，重复绝对引用中的步骤（2）、（3）、（4）、（5）；

（4）按【Alt+F11】组合键打开宏代码存放界面，在模块 2 中将看到以下代码：

```
Sub 宏 2()
' 宏 2 宏
' 以相对引用录制宏
' 快捷键: Ctrl+Shift+q
    ActiveCell.Offset(0, -1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "宏与 VBA"
    ActiveCell.Select
    Selection.Font.Bold = True
    Selection.Font.Underline = xlUnderlineStyleSingle
    With Selection.Font
        .Name = "宋体"
        .Size = 20
        .Strikethrough = False
        .Superscript = False
        .Subscript = False
        .OutlineFont = False
        .Shadow = False
        .Underline = xlUnderlineStyleSingle
        .ThemeColor = xlThemeColorLight1
        .TintAndShade = 0
        .ThemeFont = xlThemeFontMinor
    End With
    ActiveCell.Columns("A:A").EntireColumn.AutoFit
End Sub
```

由此可以明显地比较出两种引用方式录制宏代码的差异。

注意：本章学会录制宏并懂得执行宏即可，不必要纠结于能否看懂代码。在后面的章节中有关于 VBA 的基础理论介绍，当学会 VBA 后，足以编写复杂得多的代码。

本例案例文件请参考：..\ 第 1 章\ 1-3 录制宏并调用.xlsm

1.2.2　执行宏的方法

执行宏有多种方法，其中最简单的莫过于使用快捷键，即录制宏时对宏指定快捷键，录制完成后通过快捷键调用宏代码。

以上两个宏已经设置了【Ctrl+q】组合键和【Ctrl+Shift+q】作为快捷键。

通过以下步骤可执行“宏 1”。

（1）清空工作表中所有数据，便于观察执行宏之后的效果；

（2）选择 C5 单元格，按【Ctrl+q】组合键。此时光标将自动定位于 A1 单元格，且在 A1 单元格产生加粗、带下划线且字号为 20 号的字符串“宏与 VBA”，同时将 A 列调整为自动适应列宽；

（3）清除 A 列，选择 G20 单元格，再次按【Ctrl+q】组合键执行宏，可以发现两次执行的效果完全一致，运行效果如图 1-7 所示。

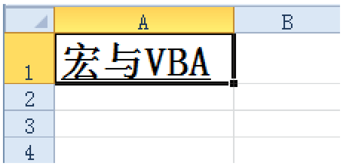


图 1–7　调用绝对引用宏的执行结果

由此可见，绝对引用的特性是总是指向固定目标，不受活动单元格位置所影响。

接下来，通过 3 个步骤调用相对引用方式录制的宏，并了解其特性。

（1）清空工作表中所有数据，便于观察执行宏后的效果；

（2）选择 C5 单元格，按【Ctrl+Shift+q】组合键，光标将自动定位于 B5 单元格，且在 B5 单元格产生加粗、带下划线且字号为 20 号的字符串“宏与 VBA”，同时将 B 列调整为自动适应列宽，运行效果如图 1-8 所示。

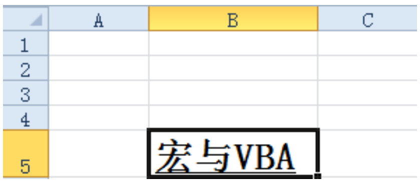


图 1–8　调用相对引用宏的执行结果

（3）选择 G20 单元格，然后再次使用【Ctrl+Shift+q】组合键执行宏，光标将自动定位于 F20 单元格，且在 F20 单元格产生加粗、带下划线且字号为 20 号的字符串“宏与 VBA”，同时将 F 列调整为自动适应列宽。

由此可以印证调用相对引用方式录制的宏时会受活动单元格位置的影响。在本例中由于录制时操作单元格在活动单元格左边的一个单元格（录制前的活动单元格是 B1，但操作单元格是 A1），所以执行宏时的操作对象总是在活动单元格左边的单元格，而非活动单元格。这个特性有时是优点，它能使宏具备更多的灵活性；有时又是缺点，调用宏时必须小心翼翼，避免操作对象错位。

1.2.3　两种方法读懂宏代码

从上面的录制宏和调用宏的过程可以初步了解宏的强大作用，那是否能读懂其代码含义呢？

完全明白所有代码的含义需要 3 到 6 个月的经验积累，而非一朝一夕之功。不过通过以下分

析有助于读者了解宏代码的大致含义。

宏代码由以下四个部分组成（见图 1-9）。

第一部分是宏的声明语句，利用 Sub 声明宏的名称，并带有一个用于存放参数的容器——空括号，例如本例中的“Sub 宏 1()”。不过录制宏时不能产生参数，使用 VBA 编程才需要参数，所以宏过程的名称只有括号，括号中总是保持空白。

第二部分是注释，它用于描述当前宏的名称、录制宏时输入的说明文字和快捷键。如果在录制宏前未添加说明，该行将显示空文本。

第三部分是宏的核心，从注释行之后直到倒数第二句的所有行，这些代码记录了录制宏过程中的所有操作信息，可以通过“播放”这些代码重现录制宏时的操作过程。

第四部分即最后一句——“End Sub”，表示结束宏过程。

其中第一和第四部分仅是一段程序的外壳，重点在于壳中的第三部分，所以大家说的“第一句代码”、“第二句代码”是针对第三部分而言，而非从 Sub 语句开始。本节中阐述的宏代码也是针对宏代码中第三部分而言。

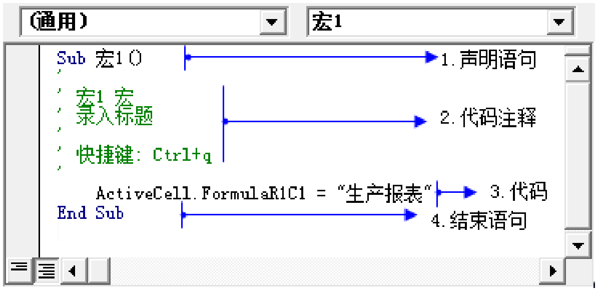


图 1-9　宏代码的结构

了解录制宏所产生的每句代码的含义，通常有两种方式：逐句执行和查询帮助。

注意：在 VBA 中，一段程序被称为过程，Sub 语句声明的程序被称为 Sub 过程，Function 语句声明的程序被称为函数过程，即自定义函数。

---

1. 逐句执行

在逐句执行代码时可以观察代码与操作对象的对应关系，了解每句代码的大致功能。

假设第一句代码是新建工作表，那么执行第一句后工作表界面会多出一个新表，根据此变化可以判断当前语句的功能是添加新表，这给新手学习代码提供较大的帮助。

以执行“宏 2”为例，读懂宏代码的方法如下。

（1）将工作表界面和代码窗口界面大小调整为各占屏幕的二分之一，方便自己能同时查看代码和工作表界面的所有可见单元格，图 1-10 即为调整好的界面；

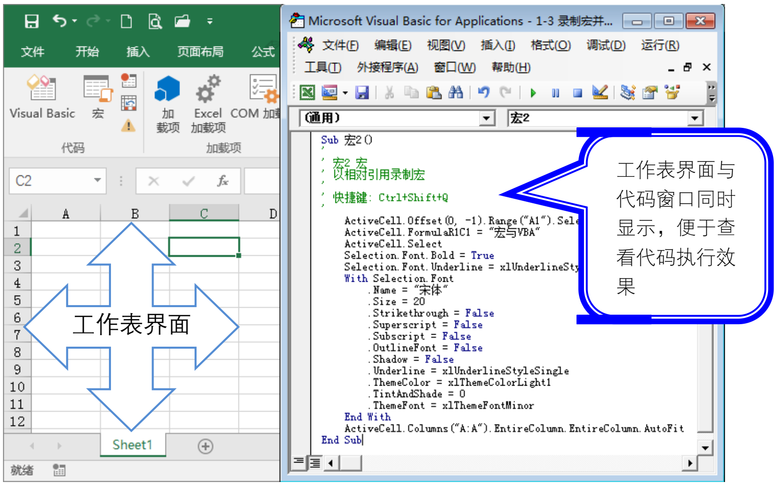


图 1-10　并排工作表窗口与代码窗口

（2）选择工作表中的 E5 单元格；

（3）激活代码窗口，用鼠标左键单击代码中任意位置，这相当于将“宏 2”设置为当前宏；

（4）按快捷键【F8】执行宏，此时 Excel 用黄色背景标示“Sub 宏 2”，表示即将执行该行代码；

（5）再次按下【F8】键，Excel 将执行“Sub 宏 2”语句，然后用黄色背景标示第二句（由于注释不属于宏代码，不需要执行，所以执行宏后会忽略。

（6）第三次按下【F8】键，Excel 才真正通过宏执行操作，即运行代码中的第一句（忽略 Sub 和 End Sub 组成的程序外壳），执行对应的功能。

当按下【F8】键后，可以发现工作表中的活动单元格已经由 F5 变为 E5，这意味着代码“ActiveCell.Offset(0, -1).Range("A1").Select”的功能是选择左边一个单元格。

（7）第四次按下【F8】键，可以发现在活动单元格 E5 中产生了字符串“宏与 VBA”，这意味着代码“ActiveCell.FormulaR1C1 = "宏与 VBA"”是对活动单元格赋值；

（8）第五次按下【F8】键，可以发现活动单元格中的字符产生了加粗效果，这意味着代码“Selection.Font.Bold = True”能对单元格的字符加粗。

按以上方式逐句执行代码，可以根据工作表的变化了解代码的含义。

然而，有时执行某些语句时可能看不到任何变化，所以此方法不能确保用户每次都能准确地猜出代码的含义。例如代码“.Strikethrough = False”的含义是去除单元格的删除线，而当单元格的字符本就不存在删除线时，屏幕上将不会有任何反应，那么也无法通过目测屏幕变化来猜测代码的含义。因此，VBA 提供了第二种协助用户读懂宏代码的方法——查询帮助。

2. 查询帮助

录制宏产生的代码大多可以通过逐句执行的方式识别每句代码的含义，然而这只是根据被操作对象的变化猜测，猜测的结果并不总是准确的，况且，如果执行某些代码后屏幕上不产生任何变化，那么将无从猜起。所以要精确地认知每句代码、每个单词的含义，可以查询 Excel 自带的帮助系统（从 Excel 2013 开始取消了本地帮助，必须在电脑联网时才能调用在线帮助）。

查询帮助也包含两种方式，一是选择需要了解的单词，例如 Offset，按下【F1】键即可。例如图 1-11 中左边部分显示了被选中的单词为 Offset，右边的帮助窗口内容便是 Offset 属性的相

关信息，包括其语法、参数含义和示例。通过帮助中的描述，足以了解 Offset 的含义及用法。

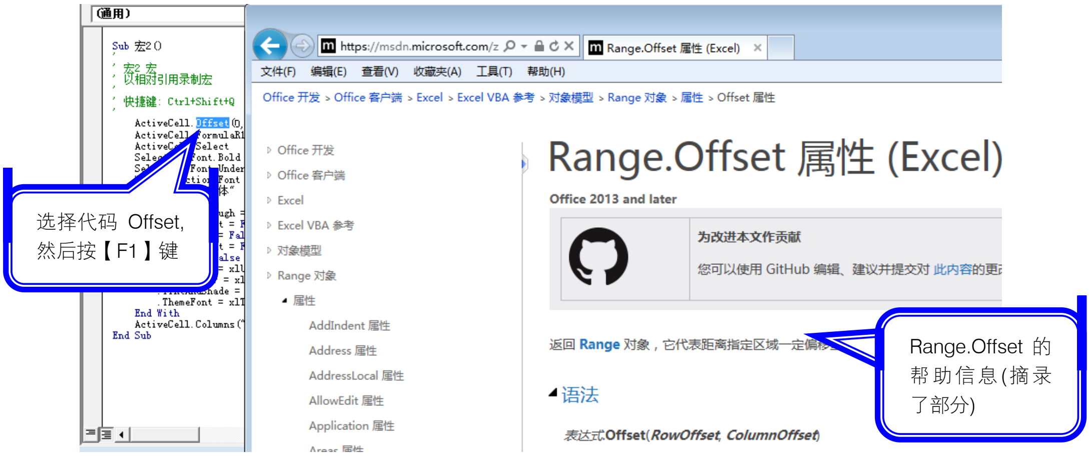


图 1–11　调用 Offset 的帮助

代码中的 Selection、Font、Bold、Select、OutlineFont 等单词都可以使用同样方法查到其含义。

1.2.4　宏的优缺点分析

根据前面三个小节，已经对宏有了基本的认识，总体来说宏有以下特性。

1．优点

（1）宏代码严格来说不是程序，录制宏及调用宏也不算编程。但是录制宏能实现与编程相近的功能，让原本需要多个步骤的工作一键完成，而且宏代码可以反复调用。

（2）录制宏时可以按操作顺序如实记录所有操作信息，所以执行宏与录制宏时的操作效果是完全一致的。而在实际工作中，当步骤较多且需要重复操作时，手动操作较难确保每次的操作顺序一致，也难确保结果的保存位置、大小等参数一致，甚至遗漏某个操作步骤的可能性也同样存在。

（3）宏是以代码的形式存在的，而代码可以调整顺序或者增删语句，所以当工作需求变化时，稍微调整宏代码即可完成所有工作，而不需要重新执行可能需要几十个步骤才能完成的工作。

（4）另外，学习录制宏仅需三分钟，在录制宏时只需要做基础操作，不需要理会代码的含义、语法、思路，宏代码就会自动生成，这较之于其他不带录制功能的编程语言有更多的优越性。

2．缺点

微软公司在 Office 平台中推广宏仅短短几年，之后就用 VBA 替代了宏的地位，无疑是宏的局限性阻碍了它的发展，而 VBA 正好可以弥补宏的缺陷，它在灵活性、效率和全面性方面都远远超过宏。

首先，并非所有操作都能通过录制宏产生相应的代码，这意味着宏的全面性不足。

其次，录制宏多数时候会产生一些冗余代码，从而降低宏的执行效率。例如前面两次录制宏时，“设置单元格的字号为 20”这一个步骤产生的代码包括设置字号、字体、删除线、阴影、主题颜色等，如以下代码所示：

```
With Selection.Font
    .Name = "宋体"
    .Size = 20
```

```
        .Strikethrough = False
        .Superscript = False
        .Subscript = False
        .OutlineFont = False
        .Shadow = False
        .Underline = xlUnderlineStyleSingle
        .ThemeColor = xlThemeColorLight1
        .TintAndShade = 0
        .ThemeFont = xlThemeFontMinor
    End With
```

如果采用 VBA 编程，那么只需要“Selection.Font.Size = 20”一句代码即可满足工作需求，所以宏与 VBA 的效率差异较大。

再次，录制宏时只能记录操作，宏代码不会执行判断。例如可以录制“删除 A 列”的操作，但是无法通过录制宏实现“假设 A 列空白则整列删除”这类需求。而且，即使是能录制的操作也仅针对单一的操作，无法录制可循环的操作。例如删除工作表 200 列数据中所有奇数列的值，如果采用录制宏产生代码，那么在录制时需要删除 100 次，产生超过 200 行的代码。如果采用 VBA 编程，那么仅需以下几句代码即可完成，而且将极大节省执行时间。

```
    Sub  删除前 200 列中奇数列的值()          '此过程代码放在模块中执行
        For i = 1 To 200 Step 2              '从 1 循环到 200，步长值为 2（即隔 2 列删除一列的值)
            Columns(i).Clear                 '将第 i 列的值删除
        Next i                               '执行下一轮循环
    End Sub
```

最后，录制宏的灵活性差。不管是单元格，还是工作表、图片、图表，在录制宏时都采用固定的名称，一旦实际情况变化时，录制宏产生的代码将会出错。例如在空白工作表中录制创建矩形并设置其格式的宏，Excel 会自动将图形命名为“矩形 1”，并对“矩形 1”设置格式。然而如果实际执行宏的环境产生了变化，宏代码在调用“矩形 1”时则可能失败。只有采用 VBA 编程才能让代码具有灵活性，自动适应环境的变化，让代码可防错，并且通用于不同环境。所以本书的重点是 VBA 教学，而不局限于应用宏。

1.2.5　如何发挥宏的长处

按前面的分析，宏能实现操作自动化，但也存在大量的问题，这些问题决定了宏无法适应工作需求中的复杂性和多变性，更不能通过录制宏产生通用的程序。然而在如今 VBA 快速发展的时代里，宏不再可能发挥其作用了吗?

其实不然，虽然在工作中 VBA 已经全方位取代了宏，但是学习 VBA 不可能迈过录制宏这道门槛。在学习过程中，每个人都要经历从低到高、从浅入深、循序渐进的阶段，总是从录制宏开始接触宏代码，并开始接触编程，然后逐步熟练掌握 VBA 的语法。

由于宏的局限性致使其失去了制表工作中的实际应用价值，但是仍然可以通过以下方法发挥宏的优势。

对于 VBA 初学者而言，可借助录制宏了解工作界面中的各种操作所对应的 VBA 代码，进而根据宏代码调用帮助查看代码的含义学习其具体的语法，从而大大提升学习进度。

在多数情况下，录制宏能产生所有需要的代码，直接使用录制宏的代码或者删除其中的冗余代码即可（冗余代码即不必要的代码，删除它也能实现同等功能）。

而对于包含循环、判断之类的工作要求，虽然录制宏不能产生循环语句与判断语句，但是可以通过录制宏产生循环与判断语句之外的代码，当录制好宏以后手动添加循环或者判断语句即可，

从而节约至少 60%的工作量。换言之，善于录制宏，可以让循环、判断语句之外的代码自动产生，而非手动录入所有代码，既提升了工作效率，又降低了出错率。

也就是说，所有能录制的操作都可以不用学习相关的 VBA 代码，录制宏可自动产生代码。需要学习的内容是不能录制的这部分，包括 Excel 的事件、条件判断语句、循环语句、防错语句、变量与常量的用法、数组和功能区设计等，而这些知识点仅占 VBA 编程的 40%左右。

与其他编程语言相比，VBA 十分简单。由于可以录制宏，还能节约至少 60%的工作量，避免花大量时间背代码，可以需要什么就录制什么，即时产生所需代码。数据有效性、排序、筛选、自定义格式、生成图表、创建工作表等都不需要消耗精力去学习和记忆，需要记的仅仅是 If Then、For Next、For Each…Next、With…End with、Do Loop、On Error Resume Next 等几组简单的语法而已。

此外，通过录制宏能快速得到程序代码，稍加修改即可正常应用，这对于新手而言极为重要，能提升自信心和成就感，而成就感是持续学习的动力。

对于 VBA 高手而言，在编程过程中同样需要录制宏。

VBA 初学者是通过录制宏来学习代码的书写方式，例如排序、筛选的代码不会写，花 10 秒钟录制宏就会了；VBA 高手则是通过录制宏快速获取代码，节约手动书写所花费的时间，也避免记错单词带来的失误，同时还可以让编程者释放背记代码的压力。由于可以录制宏，所以与工作表、单元格、图表等相关的属性、方法和事件名称统统不用记忆，在需要时录制宏即可，这让程序员从背记代码的工作中解脱出来。

总结为一句话：录制宏的目的不是使用宏，而是通过录制宏产生对应的代码，节约学习时间和书写时间，同时提升录入代码的准确性。

1.3 使用 VBA 强化 Excel 功能

想实现自动化操作，VBA 无疑是首选。当熟练使用 VBA 后，你可以让代码全自动执行，除录入数据外的所有工作都可以通过 VBA 自动完成。确切地说，VBA 是办公效率之源，能让制表文员从繁重的工作中解脱出来。

1.3.1 追根溯源：什么是 VBA

VBA 是多种应用程序通用的扩展性语言，其附加在主体程序中，为强化主体程序功能而存在。VBA 能在 AutoCAD、CorelDraw、Excel、Word、PowerPoint、WPS 等应用程序中使用，其中 Excel VBA 是指应用于 Excel 中、可扩展 Excel 功能的 VBA 编程语言，这是本书的重点。不过 VBA 不管附在何种应用程序中，其语法基础总是一致的，不同的只是应用程序的对象、属性和方法而已。所以学会了 Excel VBA 再学 Word VBA 或者 PowerPoint VBA 将得心应手、事半功倍。

Excel 为 VBA 提供了专用的界面，即 VBE 窗口（Visual Basic Editor），在工作表界面按【Alt+F11】组合键即可打开该窗口。在该窗口中包括了菜单、工具栏、工作簿与工作表对象、代码窗口等，所有与 VBA 编程相关的操作皆在此界面中完成，而调用 VBA 代码则既可在工作表界面完成也可在 VBE 窗口实现。如果代码具有自动执行功能，那么当符合条件时代码会自动启动，不需要人工干预。图 1-12 为 VBE 窗口，其中工作表、工作簿名称将随每个用户的实际环境而变化，不会完全统一。

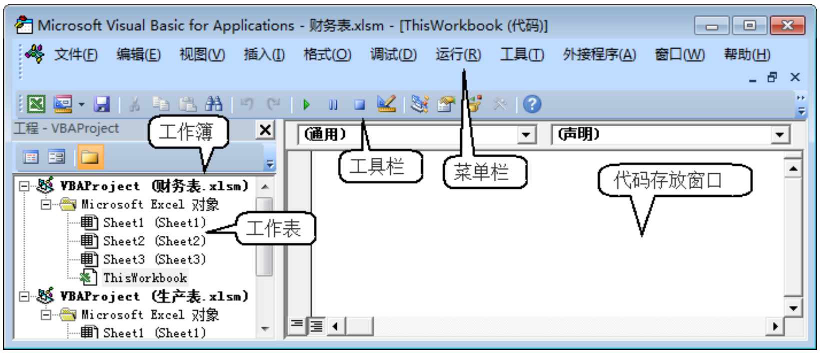


图 1–12　VBE 窗口

Excel VBA 的价值是强化 Excel 的功能，因此它只适合做与 Excel 相关的事，只能开发出 Excel 插件。如果需要开发进销存、ERP、网页系统等，则请选用其他软件。

1.3.2　知己知彼：解析 VBA 的优缺点

VBA 是微软用于取代宏的编程语言，其功能复杂、强大，能实现与制表相关的一切功能。在学习 VBA 之前，有必要了解 VBA 的优缺点。

1．缺点

Excel VBA 属于 Excel 的功能之一，在 Excel 的众多功能中，VBA 涉及的理论知识最多，它比函数、图表、透视表等工具更复杂，需要更长的时间学习，这也是众多用户对 VBA 望而生畏的原因。

好在 Excel 支持录制宏，可以节省 60%左右的工作量，极大节约了用户的时间和精力。

2．优点

相对于 Excel 的诸多其他功能而言，VBA 无疑是最强大的。VBA 能实现其他一切工具所能实现的功能，但是 VBA 能完成的工作，其他工具却不一定能实现。此处所指的“其他工具”指排序、筛选、条件格式、数据有效性、函数、图表、透视表等功能区中的常规功能。

当使用 VBA 后，可以自动化实现任何功能，而且准确、高效。例如在 A1:A10000 区域插入 10000 张图片，且将图片全部调整为所在单元格的大小，其位置刚好与单元格上边距与左边距一致，若手动实现此功能至少需要几小时，而用 VBA 可以在几秒钟内完成任务。再例如将 1000 个工作簿中的所有工作表合并到一个工作簿中，手动操作需要 1 小时以上，而且由于工作量大、步骤较多，操作过程中很难确保不会遗漏某个数据，而采用 VBA 实现同等功能写好代码仅需按下快捷键，数秒钟后即自动完成，快捷而准确。

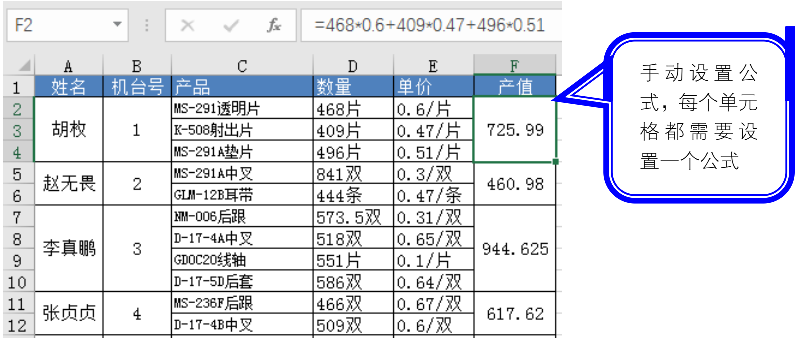
相对于其他程序语言，Excel VBA 属于最简单的编程工具，超过 60%的代码可由录制宏产生，不需要每个对象名称、属性和方法都花费精力去记忆。以 Delphi 为例，学好 Delphi 需要花费多于 VBA 三倍的精力。

1.3.3　窥斑见豹：从一个案例初识 VBA

此处以一个具体的案例展示 VBA 在工作中的应用。

图 1-13 是某公司的产值表，表中需要计算的是每个员工生产的所有产品的数量乘以单价得到的产值。要实现该需求有三个难点：每个员工生产的产品种类的数量不一致、产品数量和单价

栏中包括单位名称且单位不一致、存放结果的单元格需要合并从而无法填充公式。



姓名	机台号	产品	数量	单价	产值
胡枚	1	MS-291透明片	468片	0.6/片	725.99
		K-508射出片	409片	0.47/片	
		MS-291A垫片	496片	0.51/片	
赵无畏	2	MS-291A中叉	841双	0.3/双	460.98
		GLM-12B耳带	444条	0.47/条	
李真鹏	3	NM-006后跟	573.5双	0.31/双	944.625
		D-17-4A中叉	518双	0.65/双	
		GDOC20线轴	551片	0.1/片	
		D-17-5D后套	586双	0.64/双	
张贞贞	4	MS-236F后跟	466双	0.67/双	617.62
		D-17-4B中叉	509双	0.6/双	

图 1-13　产值计算表

该表的设计者在计算产值表时按图 1-13 中 F 列所示方式设计公式，不到 5000 行数据每天需要花费 3～5 小时才能计算完成，而且难以保障计算结果的准确度。

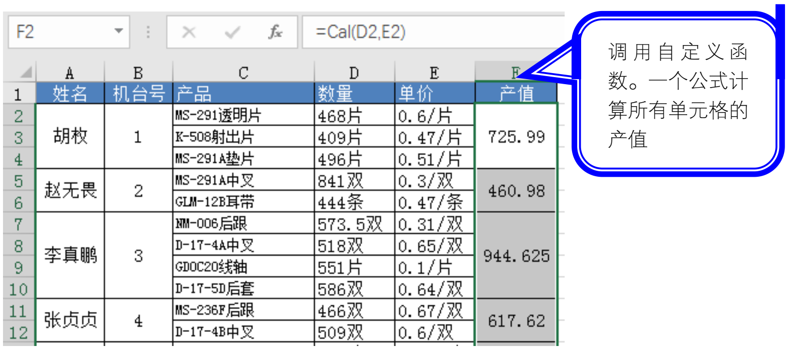
笔者为此写了一个自定义函数，只用了三句代码，3 秒钟可以完成 5000 行资料的产值计算。按以下步骤操作即可见证：

（1）打开本例案例文件：...第一章\1-4 产值计算表.xlsm；

（2）选择 F 列第 2 行到工作表最后一个非空行的整个区域（本案例文中 F2:F166 区域），然后在编辑栏中录入以下公式：

```
=cal(D2,E2)
```

（3）按【Ctrl+Enter】组合键，选区中将自动填充所有公式，整个工作 3 秒钟内完成，计算结果如图 1-14 所示。



姓名	机台号	产品	数量	单价	产值
胡枚	1	MS-291透明片	468片	0.6/片	725.99
		K-508射出片	409片	0.47/片	
		MS-291A垫片	496片	0.51/片	
赵无畏	2	MS-291A中叉	841双	0.3/双	460.98
		GLM-12B耳带	444条	0.47/条	
李真鹏	3	NM-006后跟	573.5双	0.31/双	944.625
		D-17-4A中叉	518双	0.65/双	
		GDOC20线轴	551片	0.1/片	
		D-17-5D后套	586双	0.64/双	
张贞贞	4	MS-236F后跟	466双	0.67/双	617.62
		D-17-4B中叉	509双	0.6/双	

图 1–14　使用自定义函数计算产值

本案例中使用了 VBA 开发的自定义函数 Cal，它能像所有 Excel 自带的工作表函数一样在任意单元格调用。自定义函数 Cal 的源代码如下：

```
Function Cal(Rng1 As Range, Rng2 As Range)      '放置位置：模块中
    For i = 1 To Application.ThisCell.MergeArea.Rows.Count
        Cal = Cal + Val(Rng1.Offset(i - 1)) * Val(Rng2.Offset(i - 1))
    Next
End Function
```

简单三句代码，但具有神奇的效果，用户可在 3 秒钟内完成以往 3~5 小时的工作量，这就是 VBA 的魅力。

其实 VBA 的神奇不止于此，还可以按下快捷键就完成工作。笔者对以上产值表编写了另一

段代码，并且为代码指定了快捷键【Ctrl+Shift+q】，读者可以删除 F 列的公式后再测试【Ctrl+Shift+q】组合键，你将会发现它比自定义函数更快、更方便。

具体代码在随书的案例文件中，本节仅展示 VBA 的便捷性与强大功能，在后面的章节会逐步教授编程相关的基本理论和思路，学完本书后，此案例的代码将信手拈来。

1.4　Excel VBA 的发展前景

VBA 虽然从 1993 年开始就在 Excel 5.0 中开始使用，至今已有 20 多年的历史，然而国内起步较晚，最近十多年才将 VBA 大量应用到工作中，并开始出现 VBA 专业的从业人员。

1.4.1　简化工作

Excel VBA 存在的价值在于简化制表工作，提升制表效率，同时也能确保计算的准确性。

以上一节中的自定义函数为例，它仅需三句代码，一分钟内编完程序，然后花 3 秒钟将公式写入到单元格中，可以完成原本需要 3～5 小时才能完成的工作，这无疑是 VBA 价值的体现。

此外，笔者还有一次亲身经历足以说明 VBA 的实用价值：十年前笔者曾做过半年报表汇总工作，每天将前一天的品检部、生产部和生管部报表整理到一个工作簿中，然后分组统计，并生成图表和透视表发送给经理。以前的同事每天让三个部门的文员将报表发到邮箱中，然后打开三份报表并整理数据、生成总表，该工作每天需要花费 0.5~1 小时。当笔者接手该工作后，采用以下方式处理：

（1）在服务器的共享磁盘中创建一个名为“明细”的文件夹，让三个部门的同事将每天完成的明细报表都按今天的日期命名并保存到该文件夹中；

（2）在本地磁盘创建一个文件夹“模板”，其中存放一个用于汇总的模板工作簿，该工作簿采用 VBA 代码自动打开远程共享的明细文件，并按设定的流程全自动汇总报表，当“模板”文件夹中生成汇总报表后将汇总结果自动复制一份到服务器中名为“总表”的共享文件夹中，文件以当天的日期命名。图 1-15 即为各文件夹之间的关系示意图。



图 1-15　文件夹关系示意图

（3）通知经理从服务器共享文件夹“总表”中提取今日汇总报表。

由于汇总报表中的 VBA 代码设置为打开报表时自动执行，它会到指定的文件夹中检查是否存在需要汇总的三份报表，如果不存在则产生提示信息，然后自动关闭文件；如果文件存在自动汇总报表数据，完成后在本机保存一份汇总报表，同时另存一份到共享文件夹“总表”中，一切工作全自动进行，需要执行的只有一步——双击打开本机中的汇总报表模板。换言之，以往将近 1 小时的工作现在只需要几秒钟就完成了。

后来的某天，笔者因某些原因睡过了头，第二天早上 9 点多尚未起床，结果经理 9:00 开会时需要报表。当接通经理的电话后，我在电话中告知经理汇总表已经完成，自己有事在外，并请

经理打开我的电脑，从 D 盘中打开汇总报表模板查看即可。其实当经理打开该模板时，代码才开始运行，几秒钟后自动汇总完毕，并生成一个汇总报表。

换言之，某些工作借助 VBA 可以实现无人值守，让代码全自动执行，从而解脱双手。

1.4.2 开拓专业

VBA 在中国起步较晚，目前 VBA 的从业者暂且不多，专业人员极少，且多集中在 VBA 类书籍作者、VBA 培训讲师，以及部分企业内部数据分析师（VBA 配合数据库类知识点应用）。更多人是以 VBA 技术为兼职，例如在淘宝开店为他人有偿制表、开发程序等。

不过，目前从业者少不代表开拓这个专业的机率小，反而新兴行业最有发展前途，随时可能发展出新的途径，开辟新的天地。这一切都建立在扎实的基本功上，只要努力学习 VBA 的基础理论，并多加练习，掌握处理问题的一些基础思路，技术总有转化成收益的希望，况且有录制宏的帮助，学习 VBA 必将得心应手。

1.5 课后思考

1．在录制宏时，相对引用与绝对引用的区别是什么?

2．录制宏的目的是什么?

3．在哪些情况应该使用 VBA?

4．VBA 比宏强在哪里?

5．录制宏所产生的宏代码保存在哪里? 用什么方法调用宏代码?

第 2 章　代码应用基础

在学习编写代码之前，有必要先了解如何录入、调用与保存代码，以及区分何种代码放置在何处，如何让代码畅通无阻。

本章重点在于传授 VBA 的一些周边知识，无关 VBA 语法与思路，但能借此强化基础，确保后续的学习进度与质量，提升代码录入、调用、保存方面的准确度。因此对于 VBA 初学者，本章尤其重要。

本章要点

- ◆ 区分 VBE 代码窗口
- ◆ 录入代码
- ◆ 四种代码执行方式
- ◆ 保存代码
- ◆ 让代码畅通无阻
- ◆ 反复调用相同代码
- ◆ 调用代码的帮助系统

2.1　区分 VBE 代码窗口

VBE 代码窗口是代码的载体，储存和管理代码的容器。在 VBA 中，不同的窗口中存放不同类型的代码。在使用代码前有必要先认识 VBE 有哪些代码窗口。

2.1.1　认识 VBE 窗口

VBE 全称是 Visual Basic Editor，即 VB 代码编辑器，在 Excel 中 VBE 特指保存 VBA 代码的地方。在 Excel 工作表界面按【Alt+F11】组合键或者选择菜单“开发工具”→“Visual Basic”菜单即可打开 VBE 窗口。

图 2-1 是默认的 VBE 界面，它包含了 VBA 专用的菜单、工具栏、工程资源管理器（左下角部分）、代码窗口等元素。

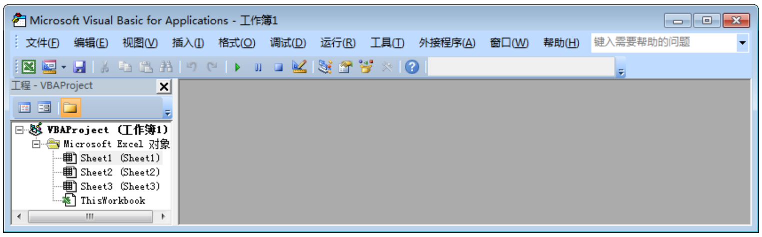


图 2–1　默认的 VBE 窗口

事实上在 VBE 中包含了更多的元素，如窗体、模块、类模块、立即窗口、监视窗口等，这

些元素默认状态是处于隐藏状态，需要用户手动调用才会显示出来。

如果需要显示模块，只需选择菜单“插入”→“模块”即可；需要显示窗体则单击菜单“插入”→“窗体”，模块、窗体以及类模块总是显示在左方的工程资源管理器中。图 2-2 体现了 VB 工程中工作簿、工作表、窗体、模块、类模块和代码窗口的默认位置关系。

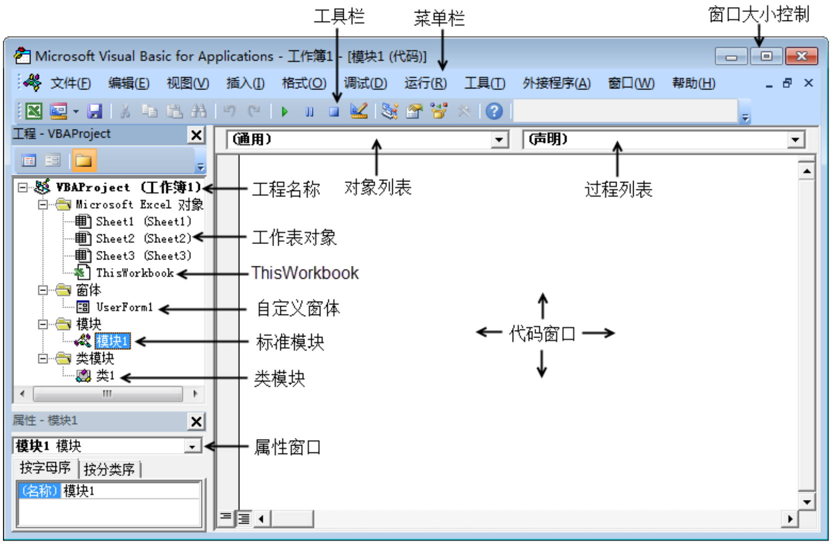


图 2–2　插入窗体、模块与类模块后的 VBE 界面

VBA 的代码窗口包括工作表事件代码窗口、ThisWorkbook 事件代码窗口、窗体代码窗口、标准模块代码窗口和类模块代码窗口。这些窗口默认是重叠显示的，也就是说显示其中一个代码窗口时其他窗口就会被当前激活的窗口所覆盖。

打开某个对象的代码窗口的方法是对该对象单击右键，在快捷菜单中选择“查看代码”，例如进入 ThisWorkbook 事件代码窗口，在左侧的工程资源管理器中右击 ThisWorkbook，从弹出的菜单中选择”查看代码”，右侧的代码窗口将切换到 ThisWorkbook 事件代码窗口。如果在进入 ThisWorkbook 代码窗口之前开启了其他代码窗口，那么该窗口将会被移到 ThisWorkbook 的代码窗口下一层。

如果工作簿中有多个代码窗口，并且需要同时查看所有代码窗口中的代码，那么可以选择菜单“窗口”→“层叠”或者“水平平铺”、“垂直平铺”。

事实上也可以双击对象打开代码窗口。例如双击 Sheet1 则进入 Sheet1 的工作表事件代码窗口；双击模块 1 则进入模块 1 的标准模块代码窗口。在图 2-3 中，双击 Sheet2 后打开 Sheet2 工作表事件代码窗口，由于未录入代码，代码窗口只有一片空白。



图 2–3　双击 Sheet2 后打开 Sheet2 代码窗口

不过，窗体对象位于右侧代码窗口的位置，和工作表、模块都不同。

选择菜单“插入”→“用户窗体”，在工程资源管理器中窗体名称默认为“UserForm1”，在右侧的代码窗口位置才是窗体对象，如图2-4所示。此时双击窗体对象即可打开窗体的代码窗口。

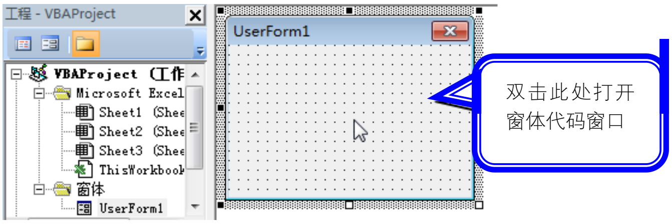


图 2-4　窗体对象的位置

注意：当工程资源管理器和属性窗口被不小心关闭后，将对编程和查看代码带来不便，可以分别采用【Ctrl+r】和【F4】快捷键将它们显示出来。

关于什么是事件，Excel 有哪些事件等问题将在本书第 3 章和第 8 章详细讲解。

2.1.2　最常用的代码存放区：标准模块

标准模块的默认名称是“模块 1”、“模块 2”等，在英文系统中则是“Module1”、“Module2”等。

一个新工作簿的工程资源管理器中默认只有工作表对象和 ThisWorkbook 对象，需要通过选择菜单“插入”手动插入模块才会显示模块和模块对应的代码窗口。

标准模块用于存放 Sub 过程(Sub 开头的过程)或者 Function 函数(即自定义函数，以 Function 开头的过程)。录制宏产生的过程都是 Sub 过程，代码保存在模块中。

假如从网上获得一段修改系统时间的 VBA 代码，如何将它应用于工作中呢？参考以下步骤：

（1）打开 Excel 软件，并使用【Alt+F11】组合键打开 VBE 界面；

（2）选择菜单“插入”→“模块”；

（3）在模块对应的代码窗口中粘贴获得的 VBA 代码：

```
Sub 修改日期()                '放置位置：模块中
  Date = #12/21/2012#         '注意：VBA 中默认的日期格式是“月/日/年”，且前后需加#
End Sub
```

（4）单击“标准”工具栏中的运行按钮执行代码（图标：▶）。

以上 4 个步骤是使用标准子过程或者宏代码的基本方法，在图 2-5 中可以看到代码的保存位置、执行代码的按钮和执行结果（注意：任务栏中的系统日期已变成 2012 年 12 月 21 日）。

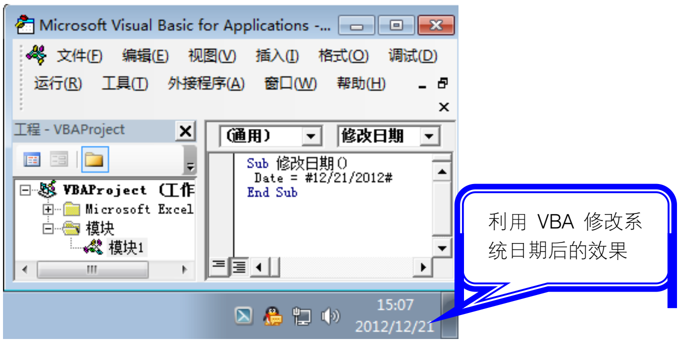


图 2-5　执行 Sub 过程修改系统日期

注意：在 VBA 中过程包括 Sub 过程（Sub 开头）、函数过程（Function 开头）和属性过程（Property 开头）。每一段完整的程序都是一个独立的过程，每录制一次宏都会产生一个过程。一个模块中可以存放多个过程，但同一个过程的代码不能放在多个代码窗口中，因为过程中的代码必须是连续的。然而 VBA 允许不同模块中的过程相互调用，即在 A 过程中可以执行 B 过程的功能。

标准子过程或者函数过程也可以保存在工作表事件代码窗口和 ThisWorkbook 事件代码窗口中，只是放在模块中更方便使用，所以建议所有 Sub 过程都保存在模块中（事件过程例外，在第 7 章、第 8 章会讲到事件过程）。

例如在一个工作簿的 Sheet1 代码窗口、Sheet2 代码窗口、Sheet3 代码窗口、ThisWorkbook 事件代码窗口和模块代码窗口中分别录入以下 5 个过程。

```
Sub A()
  MsgBox 1
End Sub
Sub B()
  MsgBox 2
End Sub
Sub C()
  MsgBox 3
End Sub
Sub D()
  MsgBox 4
End Sub
Sub E()
  MsgBox 5
End Sub
```

然后返回工作表界面，按【Alt+F8】组合键调用宏，可以看到 A、B、C、D、E 五个过程名称，其特点是模块中的过程 E 直接采用 E 即可调用，而工作表事件代码窗口和 ThisWorkbook 事件代码窗口中的过程需要添加对象名称才能调用，在使用代码调用过程时书写更烦琐。

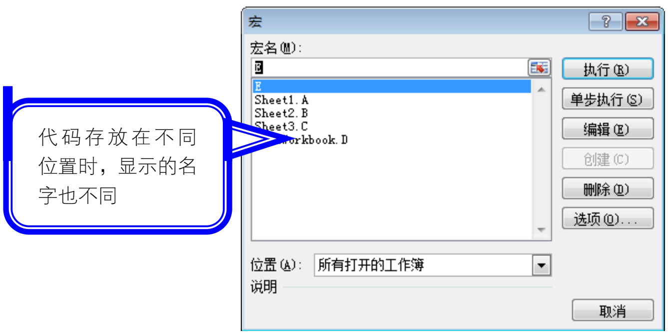


图 2-6　在不同代码窗口中的过程的显示名称

本例案例文件请参考：..\第 2 章\2-1 区分 Sub 过程存放位置.xlsm

注意：MsgBox 函数的功能是弹出一个消息对话框，并且等待用户单击其中的按钮，它同时能反馈给用户当前被单击的按钮的值。不过，虽然 MsgBox 函数能返回值，但更多的时候大家仅用它来显示信息，例如在屏幕上显示“执行完工”、“今天是 2017 年 6 月 1 日”等。

MsgBox 函数的语法如下（查询帮助的关键字：MsgBox 函数）：

```
MsgBox(prompt[, buttons] [, title] [, helpfile, context])
```

表 2-1　MsgBox函数的参数说明

参　数	说　明
prompt	它将输出到对话框中，显示在屏幕上，通常表示告诉用户某个信息。它的最大长度大约为1024个字符，由所用字符的宽度决定
buttons	它表示按钮的个数、样式和排列方式，以及默认按钮是哪一个，采用数值表示
title	在对话框标题栏中显示的字符串表达式。如果省略*title*，则将应用程序名放在标题栏中
helpfile	字符串表达式，识别用来向对话框提供上下文相关帮助的帮助文件
Context	数值表达式，由帮助文件的作者指定给适当的帮助主题的帮助上下文编号

以下代码（放置位置：模块中）是 Msgbox 函数的最常用的一种形式，它使用了 prompt、buttons、title 三个参数，不同参数决定了信息框中的不同位置的部件，效果如图 2-7 所示。

```
Sub 宏()        '放置位置：模块中
 MsgBox "您喜欢 VBA 吗?", vbYesNo, "提示"
End Sub
```

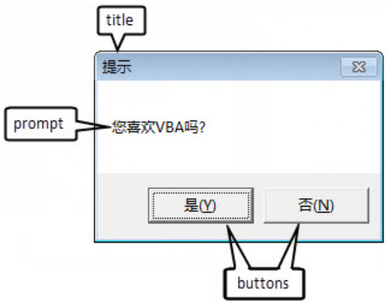


图 2–7　常见的信息框与 MsgBox 参数的对应关系

MsgBox 函数包含 5 个参数，都是可选参数。其中最重要的是第一参数，最简单的也是第一参数，第一参数所指定的字符串将显示在信息框中间。最复杂的是第二参数，它是一个数值，用于表示信息框中按钮的个数和显示样式，且能通过该数值指定默认按钮。表 2-2 展现了 MsgBox 函数所支持的按钮常数，以及对应的数值、功能和按钮示意图。

表 2-2　Msgbox第二参数的可选值说明

常　数	数　值	功　能	按钮示意图
vbOKOnly	0	只显示OK按钮	确定
vbOKCancel	1	显示OK及Cancel按钮	确定　取消
vbAbortRetryIgnore	2	显示Abort、Retry及Ignore按钮	中止(A)　重试(R)　忽略(I)
vbYesNoCancel	3	显示Yes、No及Cancel按钮	是(Y)　否(N)　取消
vbYesNo	4	显示Yes及No按钮	是(Y)　否(N)
vbRetryCancel	5	显示Retry及Cancel按钮	重试(R)　取消
vbCritical	16	显示CriticalMessage图标	
vbQuestion	32	显示WarningQuery图标	
vbExclamation	48	显示WarningMessage图标	
vbInformation	64	显示InformationMessage图标	
vbDefaultButton1	0	第一个按钮是默认值	是(Y)　否(N)　取消

续表

常　　数	数　值	功　　能	按钮示意图
vbDefaultButton2	256	第二个按钮是默认值	
vbDefaultButton3	512	第三个按钮是默认值	

对于表 2-2 需要补充以下四点。

其一，MsgBox 的第二参数可以使用表中的常数，也可以采用数值，它们功能一致。

其二，数值 0 到 5 用于确定按钮的个数和样式，16 到 64 用于确定图标的样式，256 和 512 用于表示默认按钮是第 2 个还是第 3 个。所谓的默认按钮是指单击回车键时被选中的按钮，Excel 会用特殊的外观标示这个默认的按钮。当不指定默认按钮时，第一个按钮为默认按钮。

其三，最后三行的示意图仅用于展示默认按钮的状态，三个数值的功能是指定哪一个按钮是默认按钮，而不是产生这三个图标。如果 MsgBox 第二参数采用 3+512 能产生“是”、“否”和“取消”三个按钮，那么第三个按钮是默认按钮的效果。

其四，以上 3 段代码（0～5 为第一段，16～64 为第二段，256～512 为第三段）可以相加，表示多种效果的混合应用。例如

以下两句代码能实现如图 2-8 所示效果。

```
MsgBox "VBA 很棒!", 1 + 64
```

或者：

```
MsgBox "VBA 很棒!", vbOKCancel + vbInformation
```

以下两句代码能实现如图 2-9 所示效果。

```
MsgBox "VBA 很棒?", 4 + 32 + 256
```

或者：

```
MsgBox "VBA 很棒?", vbYesNo + vbQuestion + vbDefaultButton2
```

以下代码能实现如图 2-10 所示效果（0～5 段未指定则默认当作 0 处理，即“确定”按钮）。

```
MsgBox "VBA 很棒!", vbExclamation
```

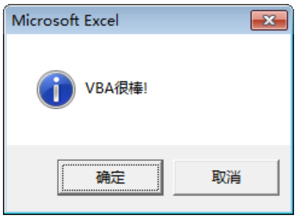


图 2-8　效果一

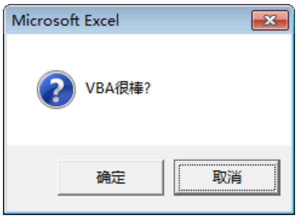


图 2-9　效果二

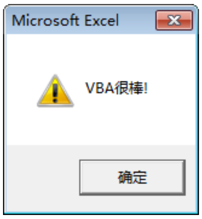


图 2-10　效果三

MsgBox 函数能弹出信息框，达到警示、通知和提示用户的作用，因此在工作中应用较广。同时，由于 MsgBox 函数具有返回值，可以根据返回值知道用户单击了信息框中的哪一个按钮，因此 MsgBox 函数也经常配合条件语句 If Then 使用，在本书后面的章节会有大量的关于 Msgbox 函数的案例。

注意：本书前 5 章属于 VBA 最基础的一些知识，在编程之前极有必要了解这些知识点。不过由于处于基础累积阶段，不适合向读者展示较复杂的案例，所以前 5 章的案例都是比较简单的一些小程序，通过这些小程序了解 VBA 的基础知识即可，不用太在意实用性，从第 6 章开始，会有大量的取材于实际工作的案例出现。

2.1.3　工作簿事件代码窗口：ThisWorkbook

ThisWorkbook 对象代表代码所在的工作簿，因此 ThisWorkbook 代码窗口也称为工作簿事件代码窗口。

ThisWorkbook 代码窗口用于保存工作簿事件的代码，本书第 7 章第 4 节会对 Excel 的工作簿事件详细介绍。

VBA 中的事件是指对象在满足指定的条件时需要做出的反应，其中工作簿事件代码指工作簿在满足代码所指定的条件时自动执行的代码。如果工作簿事件代码存放在 ThisWorkbook 以外的窗口将失去其自动性质，代码同时也失去了存在的价值。

所以对于一切工作簿事件相关的代码，都必须保存在 ThisWorkbook 代码窗口中。图 2-11 展示了工作簿的 Workbook_open 事件代码的存放方式，它的功能是打开工作簿时提示“欢迎使用VBA”。因为采用了事件，所以不需要手动执行代码，一切自动执行。

如果代码保存在其他地方则会失效。

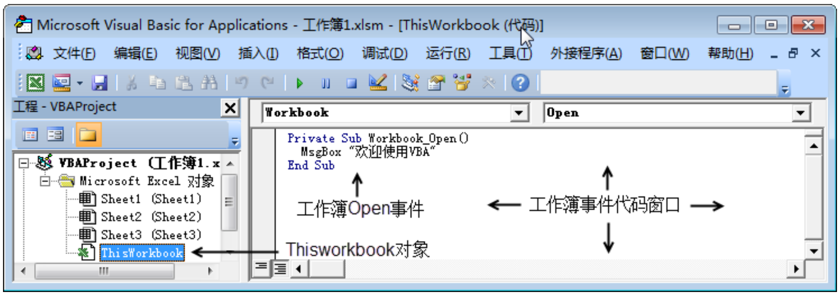


图 2–11　工作簿的 Open 事件代码及其存放位置

事件过程有别于标准过程，它的命名有相当严格的规定，过程的名字和参数差一个字母或者空格都将失去预设的功能，而标准过程的名字或者参数可以随意定义。所以通常查看一段代码的名字和参数命名方式即可判断代码是否属于事件，本书第 8 章将详细介绍事件及其应用案例。

本例案例文件请参考：..\第 2 章\2-2 工作簿 Open 事件.xlsm

2.1.4　工作表事件代码窗口：Sheet1

一个 VBA 工程中默认包含 Sheet1、Sheet2 和 Sheet3 三个工作表对象（从 Excel 2013 开始，默认只有 Sheet1 一个工作表），每个工作表各自拥有一个对应的工作表事件代码窗口。

当双击工作表对象时，在右侧会弹出该工作表对应的代码窗口。

在图 2-12 中展示了双击 Sheet2 后打开的工作表事件代码窗口，此窗口专属于 Sheet2 工作表。窗口中的代码是属于工作表级的 Change 事件代码，仅用于 Sheet2 工作表中。

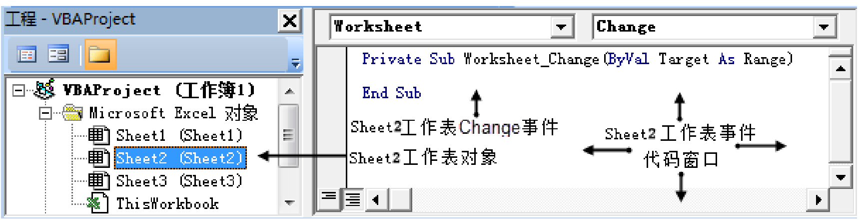


图 2–12　Sheet2 工作表的 Change 事件及其存放位置

该代码可以手动录入，也可以从上方的两个下拉列表中选择对应的值后自动生成代码。

与工作簿事件代码一样，如果将此段代码存放在其他地方则将失去其意义，不再具备 Sheet2 工作表的 Change 事件的功能，成为普通的 Sub 过程。

如果需要录入 Sheet3 工作表事件过程的代码，那么可以双击工程资源管理器中的 Sheet3，并在右侧的代码窗口中录入代码。

换言之，工作表事件代码窗口仅用于保存与工作表相关的事件过程代码。

2.1.5　窗体代码窗口：UserForm1

窗体的功能是设计对话框、制作程序界面，它的默认名称为 UserForm1、UserForm2、UserForm3 等，可以通过属性窗口修改它的名称。

本书第 12 章专门讲述窗体的应用。

VBA 工程中默认没有窗体，选择菜单“插入”→“用户窗体”，创建一个以“UserForm”加序号命名的窗体。

当插入窗体后右击窗体，选择“查看代码”，即可打开窗体代码窗口。当然也可以双击右边的窗体对象打开窗体的代码窗口，查看窗体事件的代码。

图 2-13 中展示了窗体“UserForm1”在工程资源管理器中的位置以及窗体的默认外观。当双击窗体对象后可以打开窗体事件代码窗口，便于录入该窗体的相关代码。

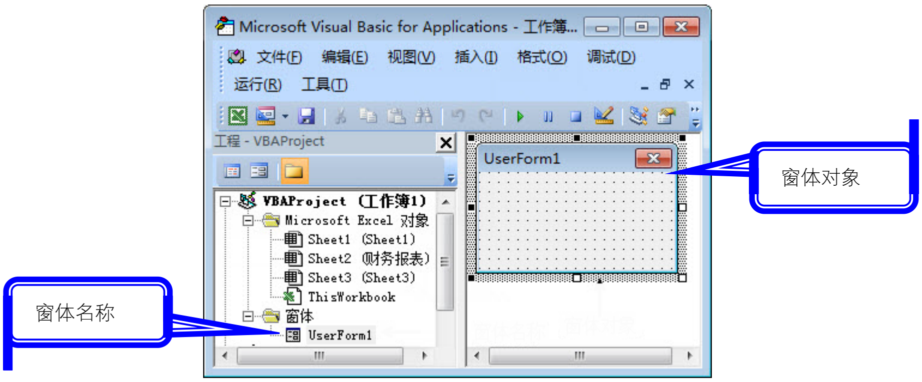


图 2-13　窗体的位置与外观

需要在窗体代码窗口录入的代码一般是窗体事件代码以及和窗体中的控件相关的代码。虽然标准过程录入到窗体中也能正常使用，但在使用时会相当麻烦。

欲了解窗体有哪些事件，请参考随书案例中的“5.窗体事件.pdf”。

本书的前 12 章不涉及窗体相关的知识，此处仅需要了解代码存放位置包含工作表事件代码窗口、工作簿事件代码窗口、窗体代码窗口和类模块即可，对窗体的功能、用法等疑问可以暂且忽略。

2.1.6　创建隐藏对象的代码窗口：类模块

类模块属于 VBA 的高级应用，需要使用类模块的情况极少。在本书第 14 章将会讲述类模块的一些简单应用。

选择菜单“插入”→“类模块”，可以在工程资源管理器中添加一个类模块，名为“类 1”，同时自动打开类模块的代码窗口。与类模块相关的代码只能录入到类模块中。

2.2 录入代码

对于 VBA 初级用户而言，应该在何处录入代码、以及以何种方式录入代码是两个最简单却又相当困扰他们的问题。这两个问题会影响代码的功能是否得以发挥，本节对此详细分析。

2.2.1 代码的存放位置

VBA 初级用户通常不是使用自己编写的代码，而是通过网络搜索他人编好的代码，或者向网友求助获得代码，然后将代码应用于工作中。然而不同的 VBA 代码需要放在不同的地方，否则可能无法正常调用代码。

根据 2.1 节的分析，所有工作表事件代码必须存放在对应的工作表的事件代码窗口中，放在其他地方代码不会自动执行；而工作簿事件代码必须保存在 ThisWorkbook 代码窗口中；类模块相关的代码必须存放在类模块中；除前面三者外的所有代码存放在标准模块中即可。

对于新手而言，用简单的一句话描述就是：除事件代码外，所有代码都适宜写在模块中。

虽然录制宏产生的 Sub 过程或者手动编写的 Function 过程也可以保存在工作表事件代码窗口以及 ThisWorkbook 窗口中，但是由于调用不方便，所以建议一律存放在标准模块中（调用不方便是指调用标准版块中的过程时只需要直接书写过程名称即可，而调用其他地方的过程则需要注明过程代码所在的位置，可参考文件“2-1 区分 Sub 过程存放位置.xlsm”）。

2.2.2 写入代码的方式

对于通过网络找到的代码或者使用网友帮忙编写好的代码，直接复制到模块中或者各类事件代码窗口中即可，本小节向读者展示的是当自己编写代码时都有哪些代码生成方式，以及在什么情况下适用哪种方式。

对于新手而言，编写代码最好的方式是录制宏，然后利用所学的 VBA 基础知识对宏代码进行修改，使其满足实际需求（录制宏通常只能产生实际需求的 30%～80%的代码，需要根据需求添加或者删除部分代码，包括添加变量，修改区域地址、路径，加入循环语句、条件判断语句或者防错语句等）。

在录制宏时，VBA 会产生操作对象的名称、属性、方法，但是不能使用变量，不能产生循环语句和条件判断语句，宏代码中的引用区域都是硬编码（例如 A1、F2:F10 等），而不能让代码的终端用户随意控制区域范围。这些问题都需要在录制宏后手动修改代码来解决，这样才能使宏代码更通用，能更灵活地适应不确定性的需求。不过总体而言，录制宏能大幅减少编程的工作量，同时也可以提升代码的准确度。

而对于录入各种事件的代码（例如工作表的 Change 事件、工作簿的 Open 事件）则应通过对象下拉列表和过程下拉列表来录入代码。因为普通的 Sub 过程和自定义函数可以随意取名，也可以随意命名参数，但是事件过程的名称和参数都有严格的规定，必须与系统内部预设的名称完全一致，所以事件的程序外壳（首尾两句）不宜手动录入，通过 VBA 提供的下拉列表选择即可，速度更快同时准确度也更高。

以录入 Sheet3 工作表的 Change 事件为例，操作步骤如下。

（1）在工作表界面按下【Alt+F11】组合键打开 VBE 窗口。

（2）如果未显示工程资源管理器则按【Ctrl+r】组合键调出工程资源管理器，如果已经显示则可以跳过本步骤。

（3）双击工程资源管理器中的工作表对象 Sheet3，然后在右边会自动出现代码窗口，

（4）单击工作表事件代码窗口上方的对象列表，从中选择“WorkSheet”，表示这是工作表对象的事件，操作界面如图 2-14 所示。

当选择对象后，在代码窗口会默认产生工作表的Change事件的过程外壳，此时可以忽略它，直接操作下一步。

（5）单击过程列表框，从下拉列表中选择 Change，VBA 会自动产生工作表的 Change 事件的程序外壳，其代码如下。

```
Private Sub Worksheet_Change(ByVal Target As Range)

End Sub
```

图 2-15 是工作表的过程列表，即工作表所支持的所有事件名称。其中 Change 事件表示当修改工作表中任意单元格的值时将执行此段代码。

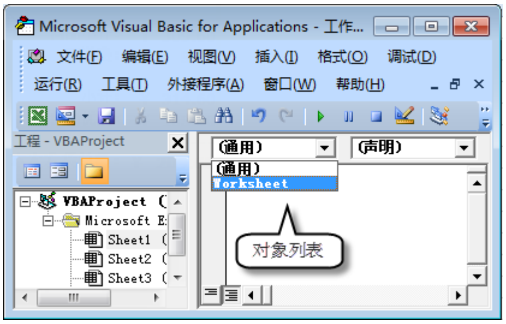


图 2–14　从对象列表选择 WorkSheet 对象

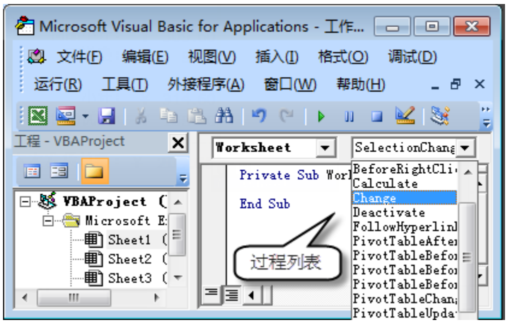


图 2–15　从过程列表选择 Change 过程名称

在选择工作表对象时，VBA 会自动产生 SelectionChange 事件的代码，此时不要删除它，当选择除 SelectionChange 外的过程名称后 VBA 会产生另一个事件的代码，此时再删除 SelectionChange 事件的代码即可。

通过此方式产生代码（程序外壳）一般不会出错，符合内部预设的规则。代码“Private Sub Worksheet_Change(ByVal Target As Range)”中的“Private”表示这是私有过程，“Worksheet_Change”则表示这是一个工作表的 Change 事件，“ByVal Target As Range”是过程的参数，代表工作表中数值正发生变化的单元格对象。在本书第 8 章中会有关于事件的详细说明，此处仅需了解事件相关的代码的录入方式即可。

对于事件过程以外的过程，VBA 提供了一个对话框帮助用户录入程序外壳（指首尾两句，对于中间的 VBA 代码一般采用录制宏或者手动编写的方式产生）。例如需要编写一个名为“合并”的 Sub 过程，可以采用以下步骤。

（1）在 VBE 界面中选择菜单“插入”→“模块”，从而新建一个模块；

（2）选择菜单“插入”→“过程”，然后在弹出的对话框中录入过程名称，并根据需求选择类型和范围，单击“确定”按钮后即可自动产生程序外壳。

图 2-16 是“添加过程”对话框，它所产生的代码如下。

```
Public Sub 合并()

End Sub
```

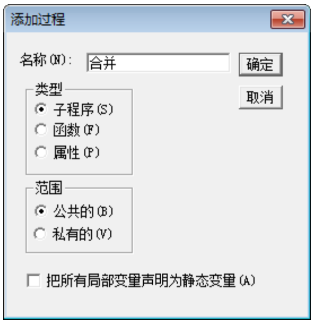


图 2-16　插入过程

如果将类型修改为函数，那么将产生以下代码。

```
Public Function 合并()

End Function
```

而当熟练掌握 VBA 语法后，可以直接手动编写代码，不需要使用此方法。

2.2.3　提升代码的可读性

在编写代码时有诸多不成文的规则，这些规则并不会强制执行，但是为了便于交流以及方便他人阅读代码，并给后续的代码维护工作提供便利，理应遵守这些规则来编写代码。

这些规则主要体现在五个方面：代码对齐、代码缩进、长代码换行、有意义的名称和注释代码。

1．代码对齐与代码缩进

VBA 代码是有层级关系的，同一级代码应采用相同的左边距，不同层级间的代码需要指定缩进距离，从而有利于阅读和理解代码。以下面这段代码为例，其具有较高的参考价值。

```
Sub 设置单元格格式()                                '放置位置：模块中
    With ActiveCell.Font                            '设置单元格的字体
        .Bold = True                                '字体加粗
        .Italic = True                              '字体倾斜
        .Underline = xlUnderlineStyleSingle         '添加单下划线
        .Name = "宋体"                              '字体名称设置为宋体字
        .Size = 9                                   '字体大小为 9 号
    End With
    With ActiveCell.Borders                         '设置单元格的边框
        .LineStyle = xlContinuous                   '边框采用实线
        .ColorIndex = 3                             '指定边框颜色为红色
        .Weight = xlMedium                          '指定线条的粗细为中等
    End With
End Sub
```

在上面的代码中，Sub 与 End Sub 属于第一层级，With 与 End With 属于第二层级，其他代码属于第三层级。第一层级代码的左边距为 0，第二层级代码的左边距为 4，第三层级代码的左边距为 8，这样的排列方式可以明显地看出各行代码之间的层级关系，将给理解代码提供较大的帮助。如果采用以下方式排列代码，则需要花费更多的时间才能理解代码。

```
Sub 设置单元格格式()                                '放置位置：模块中
With ActiveCell.Font                                '设置单元格的字体
```

```
    .Bold = True                            '字体加粗
    .Italic = True                          '字体倾斜
    .Underline = xlUnderlineStyleSingle     '添加单下划线
    .Name = "宋体"                          '字体名称设置为宋体字
    .Size = 9                               '字体大小为 9 号
    End With
    With ActiveCell.Borders                 '设置单元格的边框
    .LineStyle = xlContinuous               '边框采用实线
    .ColorIndex = 3                         '指定边框颜色为红色
    .Weight = xlMedium                      '指定线条的粗细为中等
    End With
    End Sub
```

具有层级关系的语句必须包括多行代码，在 VBA 中以下语句存在层级关系。

◆ 条件语句

```
If Then
  …
Else
  …
End If
```

这是按不同条件执行不同代码的句式，以下代码为条件语句的典型应用。

```
Sub 问候()
    If Hour(Now) > 12 Then        '如果当前时间大于中午 12 点
        MsgBox "下午好"           '弹出对话框，提示下午好
    Else                          '否则
        MsgBox "上午好"           '提示上午好
    End If                        '结束条件语句
End Sub
```

注意：VBA 中的 Now 函数表示当前系统时间，Hour 函数用于提取时间中的小时数，而 Hour(Now)嵌套应用则表示计算当前系统时间是几点钟。

关于 If Then…Else…End If 以及后面的 Select Case 语句、For …Next 语句的含义和具体用法参见本书第 6 章，此处仅需了解此类代码的层级性即可。

VBA 有几十个函数，由于逐个查询不太方便，笔者将常用的函数名称、功能、语法和应用语句整理成了一个工作簿，读者可以在需要时查阅。

VBA 常用函数请参考：..\ 附录\ 1.VBA 函数整理.xlsx

还有另一种条件语句：

```
Select Case
…
[Case Else]
…
End Select
```

方括号[]表示该部分是可选项。

◆ 循环语句

```
For
…
Next
```

或者

```
For Each
…
Next
```

或者

```
Do
  …
[Exit Do]
  …
Loop
```

或者

```
While
…
Wend
```

◆ With 语句

```
With
…
End With
```

◆ GoSub 语句

```
GoSub line
...
line
...
Return
```

对于以上代码的用法和含义将在本书第 6 章和第 9 章进行详解。

本例案例文件请参考：..\ 第 2 章\ 2-3 提升代码的可读性.xlsm

2．长代码换行

当一段代码中有一行或者多行长代码时，需要反复拖动滚动条才能查看完整代码，不利于阅读和理解代码含义。

VBA 提供了代码换行的方式：在代码中需要换行的地方插入“_”，然后将它后面的代码放在下一行即可，从而使单行代码显示为两行或者多行。

要注意的是“_”包含两个字符，第一个字符是空格，第二个字符是下划线。

以下过程只有一句代码，表示提取 C 盘的盘符、序列号、总空间和剩余空间。由于该行代码过长，任何显示器都无法完整显示。

```
Sub 磁盘信息 1() '一句代码获取磁盘的盘符、序号、总容量与剩余空间
    MsgBox "盘符：" & CreateObject("Scripting.FileSystemObject").GetDrive("C:\").DriveLetter & Chr(10) & "序号：" & CreateObject("Scripting.FileSystemObject").GetDrive("C:\").SerialNumber & Chr(10) & "容量：" & CreateObject("Scripting.FileSystemObject").GetDrive("C:\").TotalSize / 1024 & Chr(10) & "剩余空间：" & CreateObject("Scripting.FileSystemObject").GetDrive("C:\").FreeSpace / 1024
End Sub
```

如果将代码截断为多行，使其完整地显示在一屏中将对阅读代码提供便利，那么在需要截断的代码处添加“_”，换行前后的效果如图 2-17 如示：



图 2–17　长代码换行与不换行的效果比较

在换行时需要注意的是不能将代码从单词中间截断，即单个单词不能显示在两行中，否则代码无效。

鉴于本例代码的特殊性，也可以使用 With 语句来简化代码，改为后代码如下所示。

```
Sub 磁盘信息 3()
    With CreateObject("Scripting.FileSystemObject").GetDrive("C:\")
        MsgBox "盘符：" & .DriveLetter & Chr(10) _
            & "序号：" & .SerialNumber & Chr(10) _
            & "容量：" & .TotalSize / 1024 & Chr(10) _
            & "剩余空间：" & .FreeSpace / 1024
    End With
End Sub
```

本例案例文件请参考：..\ 第 2 章\ 2-4 长代码换行与不换行效果比较.xlsm

3．有意义的名称

名称包括程序过程的名称和代码中使用的变量名称。为了确保代码的可读性，便于自己或他人理解代码，有必要给过程和变量取个有意义的名称。

例如需要编写一个合并 D 盘下“生产表”文件夹中所有工作簿数据的过程，那么可按以下两种方式编写。

```
Sub 合并 D 盘生产表文件夹所有工作簿()
 …过程代码…
End Sub
Sub 合并工作簿() '将 D 盘中“生产表”文件夹中所有工作簿的数据合并到一个工作簿
 …过程代码…
End Sub
```

也就是说要么在命名时完整说明此段代码的功能，要么在命名时简化说明，并且在后面追加详细注释。但是在命名时也需要指出程序的大致功能，绝对不能用 A 或者 B 对过程命名。

对于过程中需要使用的变量或者常量名称也一样，不能使用 A、J、K、X、Y 之类的字母。如果对自己的英文水平自信，那么可以用英文对变量命名，例如根据“Dim FileName as String”中的“Filename”变量名称可以判断这个变量用于表示文件名称。也可以采用汉字词语对变量或者常量命名，例如“Dim 文件名 as String”。这两种方式都有助于快速理解代码。

关于什么是变量、常量以及 Dim 语句的用法请参考本书第 5 章。

4．注释代码

注释代码应该成为日常工作习惯，既方便他人也方便自己，任何时候打开代码都能迅速明白代码的含义、编写思路和代码的功能。

代码注释包括两种，一是整段代码的功能描述，通常写在过程的顶端；二是对每句代码添加含义说明，通常写在代码的右侧。如果代码较长也可以将注释写在代码的上方。

添加注释的方法是先在半角状态下录入撇号“' ”，然后录入注释内容，注释内容不会被执行。VBA 默认采用绿色标示注释内容，便于用户区分代码和注释。

在以下代码中前两句为注释，用于说明 Sub 过程的功能，在每句代码的右侧也有相应的注释，表示代码的含义。这种代码编写方式更有利于理解和维护代码。

```
'功能：打开 Excel 时自动问候，不需要手动执行
'如果当前时间大于 12 点则提示下午好，否则提示上午好
Sub Auto_open()                 '过程名为 Auto_open 表示此过程能自动执行
    If Hour(Now) > 12 Then      '如果当前时间大于中午 12 点
        MsgBox "下午好"          '弹出对话框，提示下午好
    Else                        '否则
        MsgBox "上午好"          '提示上午好
    End If                      '结束条件语句
End Sub
```

本例案例文件请参考：..\第 2 章\2-5 对代码添加注释.xlsm

2.2.4　调用快速信息

录入代码时有诸多技巧，掌握技巧可以简化录入工作，同时也能确保代码的准确性。录入代码的技巧之一是录制宏，在前面已经反复地强调过，其二是调用快速信息。

VBA 对很多对象都提供了快速信息，快速信息包括对象、方法、属性、函数和参数等 VBA 相关的重要信息。图 2-18 和图 2-19 分别是单元格对象 Range 和工作表对象的快速信息，其中绿色图标代表方法，黑色带有手形的图标代表属性。如果对这些方法或者属性的写法把握不准时，从快速信息列表中选择现成的名称能提升录入效率，同时也能确保准确度。

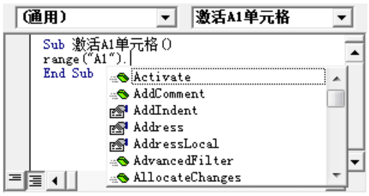


图 2-18　Range 对象的快速信息

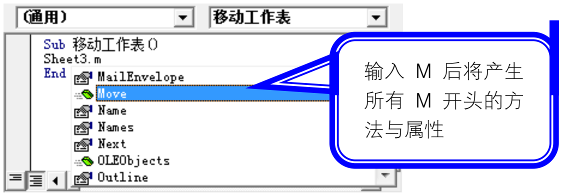


图 2-19　工作表对象的快速信息

事实上，VBA 还提供了更多的快速信息帮助用户学习 VBA，例如输入“vba.”，立即产生 VBA 所支持的所有函数名称和常数，如图 2-20 所示。

正是基于这种便利性，学习 VBA 的速度可以提到明显提升，很多原本需要记忆的对象、属性或者函数名称都不再需要记忆，或者说不需要完整地记下来。例如某用户仅记得工作表函数中的转置函数单词较长，以 Tr 开头，无法完整地拼写出单词，但利用 VBA 的快速信息功能却可以准确并且快速地录入该函数名称，具体操作如图 2-21 所示。

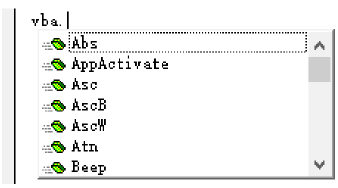


图 2-20　VBA 对象的快速信息



图 2-21　通过快速信息录入函数名称

其他函数或者对象名称也一样，只要有一点印象即可，而不需要完整记下整个单词。当然，以 Tr 开头的单词不只一个，但是 VBA 提供了【F1】快捷键查看单词的含义与语法，从而帮助了解快速信息中的每个单词的功能和用法。因此掌握学习 VBA 的方法远比记忆能力更重要。

2.3 四种代码执行方式

不管以何种方式获得代码，首要问题都是代码放在何处以及如何调用这段代码，否则代码只是一些字母的组合而已，无法对工作带来帮助。

2.2 节阐述了代码的存放区域和录入方式，本节展示调用代码的四种方式。

2.3.1 调用快捷键

Excel 的“宏”对话框可以对 Sub 过程指定快捷键。假设已有名为“问候”的程序代码，对它指定快捷键并调用的步骤如下。

（1）在工作表界面使用【Alt+F11】快捷键打开 VBE 界面；

（2）选择菜单“插入”→“模块”，然后将获取的代码粘贴在模块中。

```
Sub 问候()
    If Hour(Now) > 12 Then          '如果当前时间大于中午 12 点
        MsgBox "下午好"              '弹出对话框，提示下午好
    Else                            '否则
        MsgBox "上午好"              '提示上午好
    End If                          '结束条件语句
End Sub
```

（3）返回工作表界面，使用【Alt+F8】组合键打开“宏”对话框。由于列表中只有一个宏，所以默认选中的宏名即为“问候”；

（4）单击“选项”按钮打开“宏选项”对话框，在快捷键文字框中录入小写字母 q，表示对宏指定快捷键为【Ctrl+q】。图 2-22 和图 2-23 分别为“宏”对话框和指定快捷键的界面。

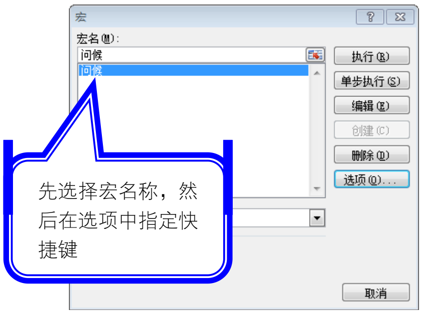


图 2–22 “宏”对话框

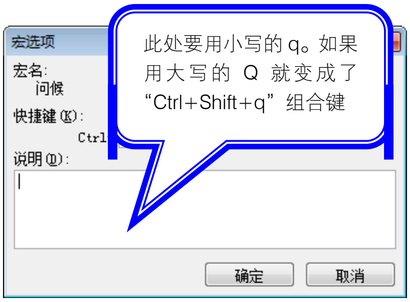


图 2–23 对宏指定快捷键

（5）返回工作表界面，按下【Ctrl+q】组合键，此时会发现程序“问候”立即执行。如果当前时间大于 12 点则提示“下午好”，否则提示“上午好”。

VBA 还提供另一种方式对 Sub 过程指定快捷键——OnKey 方法。

下面这段代码表示对过程“问候”指定快捷键【Ctrl+Shift+q】，其中“^”代表【Ctrl】键。

```
Sub 设置快捷键()                      '放置位置：模块中
 Application.OnKey "^Q", "问候"      '当前指定的快捷键为[Ctrl+Shift+q]
End Sub
```

当执行以上代码后，快捷键【Ctrl+Shift+q】即与过程“问候”捆绑在一起，在返回工作表界面后可以通过此快捷键调用程序。要注意的是快捷键必须在工作表界面使用，不能在 VBE 界面中使用。

如果需要指定快捷键【Ctrl+q】，那么 onkey 的第一参数改为“^q”即可。

本例案例文件请参考：..\ 第 2 章\ 2-6 使用快捷键调用过程.xlsm

2.3.2　单击按钮执行

使用快捷键执行过程虽然较快，但是通常只是方便记住自己设置的快捷键，如果把工作簿发送给他人使用，则会带来不便。另外，如果有多个过程指定了多个快捷键就更容易出错了。基于此问题，将过程关联到按钮，并统一按钮名称与过程名称，则既可以防止出错，又可以避免花费精力去记忆快捷键。

将一个过程关联到按钮的步骤如下。

（1）在 VBE 界面选择菜单“插入”→“模块”。

（2）在模块中录入以下代码。

```
Sub 问候()                          '放置位置：模块中
    If Hour(Now) > 12 Then          '如果当前时间大于中午 12 点
        MsgBox "下午好"              '弹出对话框，提示下午好
    Else                            '否则
        MsgBox "上午好"              '提示上午好
    End If                          '结束条件语句
End Sub
```

（3）返回工作表界面，选择功能区中的“开发工具”→插入“按钮（窗体控件）”，然后在工作表中拖放，从而绘制一个按钮。绘制好按钮后，Excel 会弹出“指定宏”对话框。

（4）在宏名称列表中选择“问候”，然后单击“确定”按钮，界面如图 2-24 所示。

（5）返回工作表界面，右击按钮，从快捷菜单中选择“编辑文字”，将默认的“按钮 1”修改为“问候”；

（6）单击“问候”按钮（见图 2-25），如果当前时间大于 12 点则提示“下午好”，否则提示“上午好”。

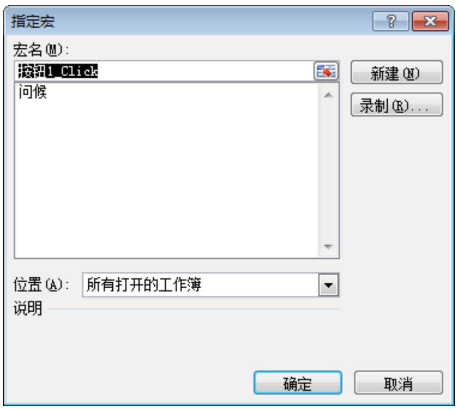


图 2–24　为按钮指定宏

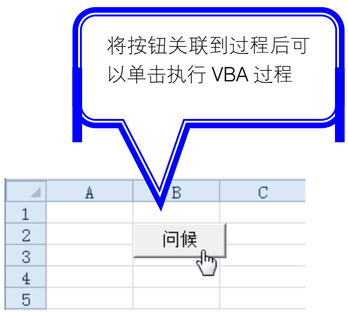


图 2–25　通过按钮执行代码

本例案例文件请参考：..\ 第 2 章\ 2-7 将过程关联到按钮.xlsm

注意：使用快捷键和按钮调用过程仅对无参数且未隐藏的 Sub 过程生效，对自定义函数无效，对有参数的 Sub 过程无效，对通过 Private 声明为私有过程的 Sub 过程也无效。

2.3.3 自动执行

在 VBA 中有一种特殊的过程——事件过程。使用快捷键或者按钮都无法调用事件过程，事件过程只有在符合事件所指定的条件时自动执行，它不需要人工干预。

事件过程包括三项内容：触发事件的对象、触发事件的条件和触发条件后执行的命令。其中前两项由事件过程的名称和参数决定，即由 VBA 预先设置。第三项由用户手动录入，可以是执行任意操作的命令代码。

例如在 Worksheet_SelectionChange 事件过程代码中，触发事件的对象是 Target 参数，Target 代表当前选中的区域；触发事件的条件由过程名称决定，即“Worksheet_SelectionChange”，它表示在工作表中选择单元格时触发条件；而触发条件后执行的命令是代码“MsgBox Target(1).Value”，它表示将选区中第一个单元格的值显示在信息框中。

```
'放置位置：Sheet1 代码窗口中
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
 MsgBox Target(1).Value
End Sub
```

上述代码的具体使用方法如下：

（1）在工作表界面按【Alt+F11】组合键打开 VBE 窗口；

（2）在工程资源管理器中双击工作表 Sheet1，在右边的代码窗口中录入或者粘贴以上代码。

（3））返回工作表界面，进入 Sheet1 工作表，选择任意区域或者单元格，Excel 都会弹出对话框提示选择区域中左上角单元格的值。如果选择了空白区域则提示空文本。图 2-26 为选择 A3:B4 区域后程序自动执行的结果。



图 2-26　工作表 Selection-Change 事件执行结果

如果需要在遇到空白单元格时不产生提示，那么代码可以改为以下内容。

```
If Len(Target(1)) > 0 Then MsgBox Target(1).Value
```

上述代码的含义是如果选区左上角单元格的长度大于 0，那么提示该单元格的值。这样指定条件后将会忽略空单元格。

此案例中“Worksheet_SelectionChange”过程完全是自动执行的，不需要每更新一次选择区域就手动调用一次过程，这也是事件过程的特点。

本例案例文件请参考：..\ 第 2 章\ 2-8 Worksheet_SelectionChange 事件.xlsm

再看看另一种事件的执行方式——工作 Open 事件，即打开工作簿时执行的事件。

（1）在工作表界面使用【Alt+F11】组合键打开 VBE 窗口；

（2）在工程资源管理器中双击“ThisWorkbook”，在右侧的代码窗口中录入以下代码；

```
Private Sub Workbook_Open() '放置位置：ThisWorkbook 事件代码窗口中
 MsgBox "欢迎使用 Excel，今天是" & Date
End Sub
```

此代码表示打开工作簿时提示今天的日期。代码中的“Workbook”表示它是一个工作簿事件，所以代码存放在 ThisWorkbook 代码窗口中；“Open”表示打开工作簿时触发事件，而 Date 函数表示获取今天的日期，所以整个过程的含义是打开工作簿时报告今天的日期。

（3）将工作簿另存为“Workbook_Open 事件.xlsm”，然后重启此工作簿，在打开工作簿后将自动弹出图 2-27 所示的信息框。

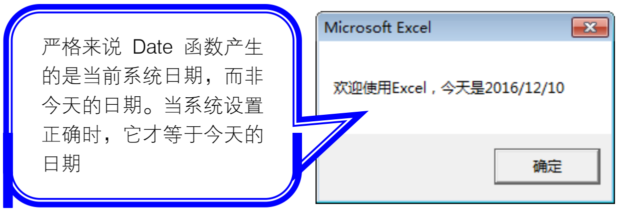


图 2–27　自动产生的日期提示

根据以上两个案例可以窥斑见豹，对事件类过程有初步的认识和了解。在本书第 8 章将会详细剖析事件的分类和应用。

本例案例文件请参考：..\ 第 2 章\ 2-9 Workbook_Open 事件.xlsm

2.3.4　在公式中调用

VBA 还有一种 Function 过程，即自定义函数。此过程可以在 VBA 中嵌套到 Sub 过程中，即用 Sub 过程调用 Function 过程。但是该过程更多的是在单元格中被当作公式来运用。

以自定义函数“及格率”为例，使用步骤如下。

（1）在工作表界面按【Alt+F11】组合键打开 VBE 窗口。

（2）选择菜单“插入”→“模块”，然后在右侧的代码窗口中录入以下代码。

```
'自定义函数，名称为“及格率”，有一个 Range 型的参数，参数必须是单元格引用
Function 及格率(cell As Range)  '放置位置：模块中
 '用大于等于 60 的个数除以数字个数，即得到及格率。并将它赋值给函数。
 及格率 = WorksheetFunction.CountIf(cell, ">=60") / WorksheetFunction. CountIf(cell, ">0")
 '再通过 Format 函数将产生的小数转换成百分数，且保留两位小数
 及格率 = Format(及格率, "0.00%")
End Function
```

代码中的注释部分可以忽略不录入，它不影响代码的运行效果，但是应该养成对代码添加注释的习惯。

（3）返回工作表界面，在 E2 单元格录入以下公式，公式结果如图 2-28 所示。

```
=及格率(B2:C11)
```

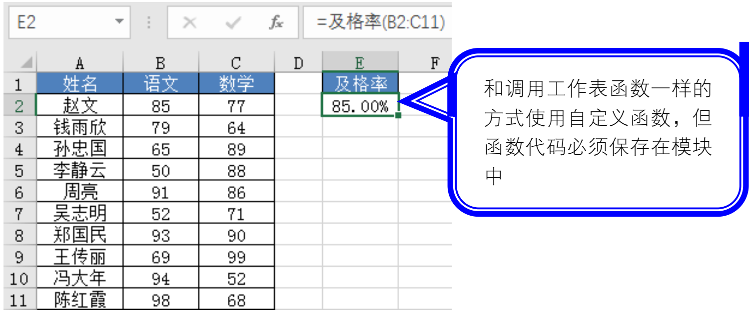


图 2–28　计算及格率

可见 Function 过程可以和内置的工作表函数一样在单元格中使用，它能简化公式，且让用户更容易看懂公式，同时也简化了录入公式的工作。

本例案例文件请参考：..\ 第 2 章\ 2-10　自定义函数.xlsm

2.4　保存代码

对于 VBA 初学者来说，一定遇到过保存工作簿后再次打开文件时发现代码已消失的问题，本节讲述如何保存有 VBA 代码的工作簿，以及如何单独保存代码。

2.4.1　修改文件的保存格式

从 Excel 2007 开始，需要将包含宏代码和不包含宏代码的工作簿采用两种格式加以区分，前者使用 xlsm 格式，后者采用 xlsx 格式。

如果工作簿中包含 VBA 代码或者宏表函数，且在保存工作簿时选择了 xlsx 格式，那么 Excel 会弹出如图 2-29 所示的对话框，表示 xlsx 格式的文件不能保存代码。

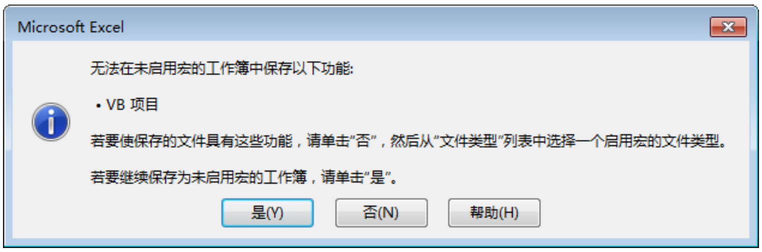


图 2–29　将包含宏的工作簿保存为 xlsx 格式时的提示对话框

如果需要保留工作簿中的代码，那么此时可以选择“否”，然后重新从保存类型列表中选择格式为“Excel 启用宏的工作簿(*.xlsm)”，操作界面如图 2-30 所示。

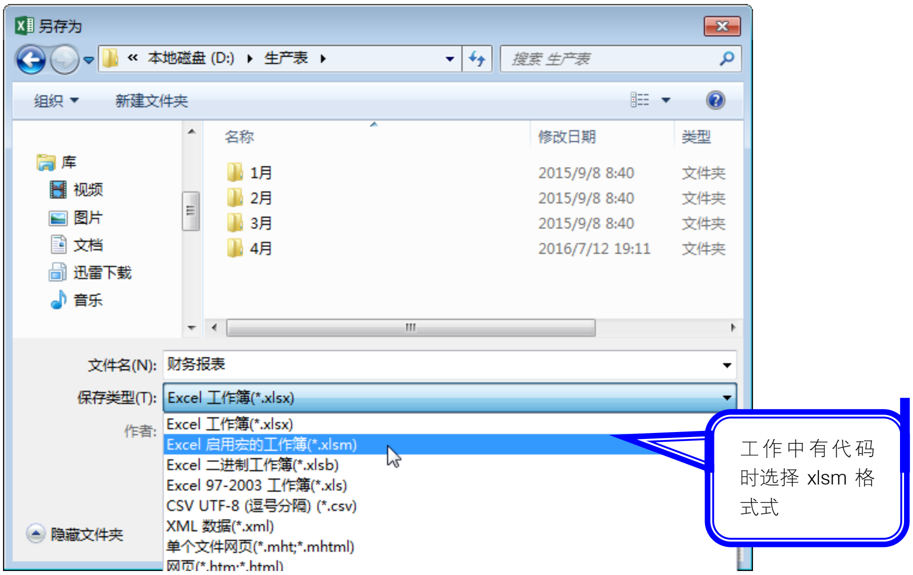


图 2–30　保存带 VBA 代码的工作簿时选择 xlsm 格式

2.4.2　一劳永逸

前面的方法是将当前保存的文件修改为 xlsm 格式，从而避免工作簿中的 VBA 代码自动消失。如果要一劳永逸，让每次保存文件时都默认采用 xlsm 格式，那么可以通过修改 Excel 选项来完成，方法如下。

（1）按【Alt+t+o】组合键打开“Excel 选项”对话框；

（2）在“保存”选项卡的“将文件保存为此格式”列表中选择“Excel 启用宏的工作簿(*.xlsm)”，表示后续的所有文件都默认保存为启用宏的格式，从而避免丢失代码，如图 2-31 所示。

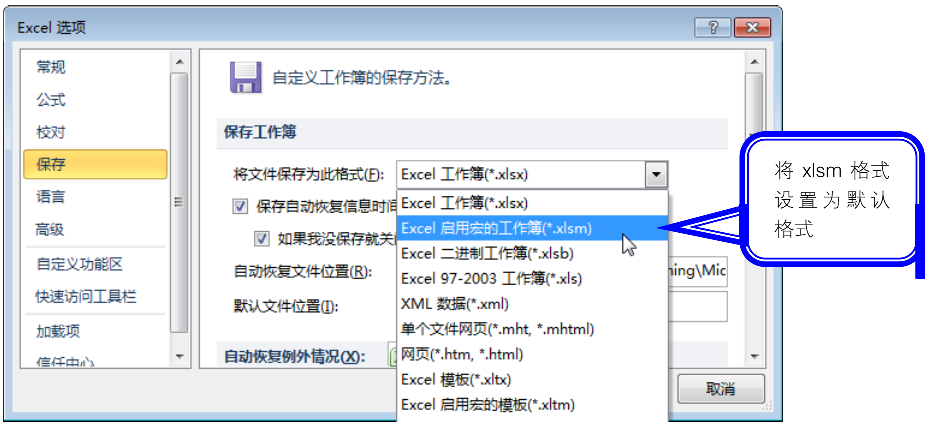


图 2–31　将文件的默认格式修改为 xlsm 格式

2.5　让代码畅通无阻

在 20 世纪 90 年代，由于宏代码十分强大而且普及率高，使得宏病毒泛滥，微软为了防止大量用户被宏病毒困扰，将宏的安全性提高了，在默认设置下将阻止宏代码运行。然而这也导致了

VBA 用户需要手动调整选项，否则便无法使用宏。

2.5.1　调整宏的安全等级

Excel 2016 将宏的安全性设为 4 级，默认是第 2 级——禁用所有宏，并发出通知。这表示默认状态下所有宏都不会执行，Excel 会弹出安全警告，通知用户当前宏已被阻止。图 2-32 是宏设置的可选项，而图 2-33 则是宏代码被阻止时的安全警告。

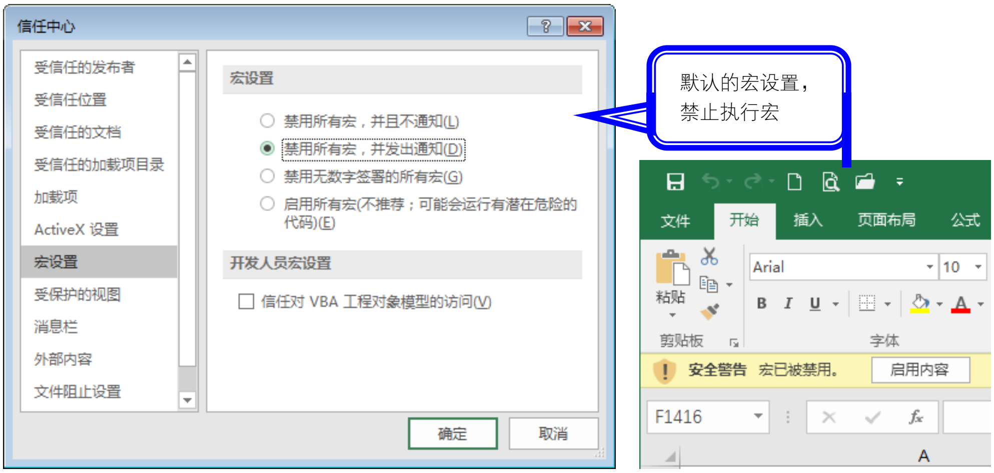


图 2-32　宏设置的 4 个可选项　　　　图 2-33　安全警告

当出现如图 2-33 所示的安全警告时，工作簿中的一切代码都禁止运行。如果需要本次打开工作簿时运行工作簿的代码，那么单击“启用内容”即可。

如果需要让所有工作簿中的代码都能顺利执行，而不是每次手动单击“启用内容”，那么可以调整宏设置，从而一劳永逸，操作步骤如下。

（1）选择菜单“开发工具”→“宏安全性”，打开设置界面。

（2）将宏设置修改为第 4 项——启用所有宏，界面如图 2-34 所示。

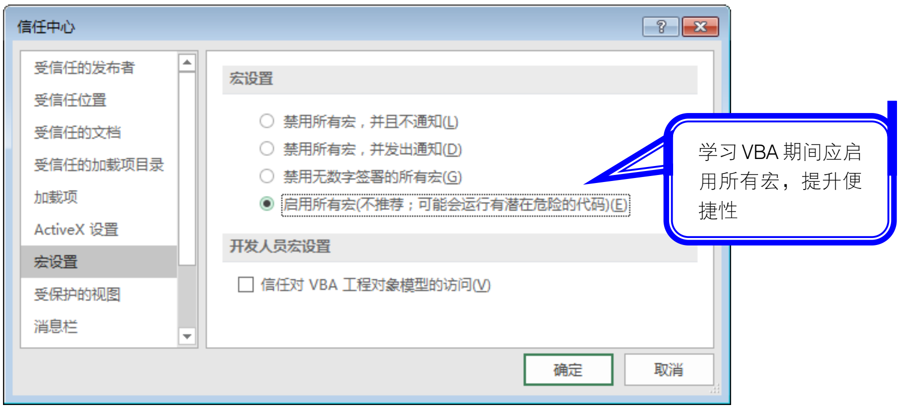


图 2-34　修改宏设置

2.5.2　添加受信任位置

如果说前面的方法有安全隐患，会造成安全与不安全的代码都自动运行，那么添加受信任位置则可以面面俱到——既能避免不信任的文件自动启用宏代码，又能确保已确认代码安全的工作

簿可以正常执行宏，方法如下。

（1）将宏设置恢复到默认状态——禁用所有宏，并发出通知。

（2）选择菜单“开发工具”→“宏安全性”，打开 Excel 的信任中心。

（3）选择对话框中的“受信任位置”，并单击右方窗口中的“添加新位置”按钮，打开“Micosoft Office 受信任位置”对话框。

（4）单击“浏览”按钮，然后将需要信任的文件夹添加到对话框中。如果该文件夹的子文件夹也需要受信任，可以将“同时信任此位置的子文件夹”打勾。具体操作步骤如图 2-35 所示，图中的操作结果为 D 盘中的“财务报表”及其子文件夹皆受信任。

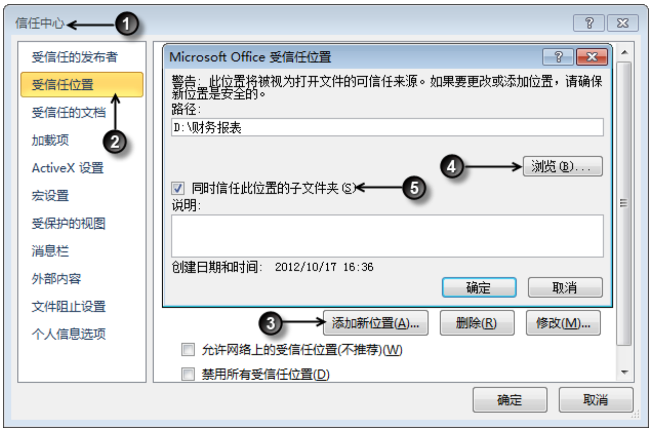


图 2-35　添加受信任位置

经过以上设置后，只要将带有宏代码的工作簿保存在 D 盘“财务报表”文件夹中就可以在打开文件时正常执行宏，而其他路径下的文件则默认禁用宏，并发出通知。

2.5.3　将代码封装为加载项

除上面两种方法能让代码畅通无阻地运行外，还有第三种方式——将宏代码封装成加载项，即 DLL 动态链接库文件。

Excel 2016 信任中心的宏设置仅对 Excel 工作簿中的 VBA 代码生效，对于 DLL 格式的加载项无效，所以代码封装成加载项后可以自动执行，不受宏安全性限制。

封装 VBA 代码属于 VBA 的高级应用，必须熟练掌握好 VBA 的基础理论后再踏入该知识领域。本书的最后几章会详细阐述如何封装代码。

2.6 反复调用相同代码

VBA 代码可以反复调用，也可以让同一段代码在不同工作簿中都能调用，从而节约编写代码的时间。本节介绍三种方法让代码可以反复调用。

2.6.1　使用个人宏工作簿

经常录制宏的读者往往会对个人宏工作簿有较多的了解，个人宏工作簿名为 PERSONAL.XLSB，保存在自启动路径中，随 Excel 软件开启而自动打开。如果在个人宏工作簿中有 VBA 代码，那么该代码可以在当前打开的任意工作簿中按下【Alt+F8】组合键调用。

向个人宏工作簿中写入代码有两种方式：录制宏和手动复制代码到 PERSONAL 文件。

1．录制宏

在录制宏时，如果将“保存在”设置为“个人宏工作簿”（见图 2-36），那么 Excel 会立即创建名为 PERSONAL 的文件，然后将录制宏所产生的代码自动保存在该文件中。以后不管任何时候打开任何工作簿，个人宏工作簿都会自动启动，并且能通过【Alt+F8】组合键调用其中的所有宏。

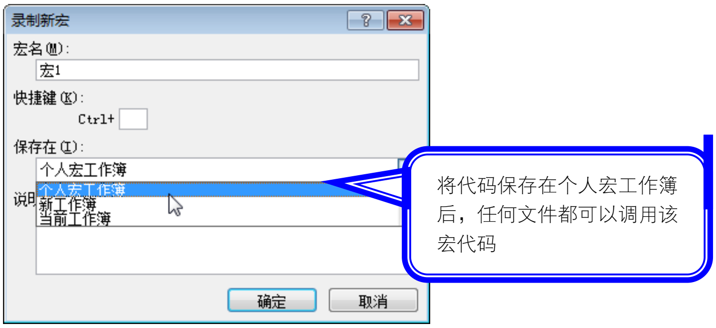


图 2-36 让录制宏的代码保存在个人宏工作簿

2．手动复制代码到 PERSONAL 文件

录制宏有诸多限制，不能录制自定义函数，也不能录制带参数的过程以及判断语句、循环语句、防错语句等，对于这类更复杂的代码只能进入 PERSONAL 文件，然后手动录入，或者在其他工作簿中编写代码并测试好后复制到 PERSONAL 文件的模块中。

不过 PERSONAL 文件需要录制宏才产生，所以可以随意录制一个宏，将它保存在个人宏工作簿中，然后在 VBE 界面打开 PERSONAL 文件，将新的代码覆盖宏代码即可。

注意：如果在录制宏时代码保存在当前工作簿，那么该宏只能打开本工作簿时才能使用，当关闭工作簿后打开其他文件时将无法调用宏。

2.6.2 加载宏

加载宏是一种特殊的工作簿，通常为 xla 和 xlam 格式。加载宏类似于游戏的外挂，用于强化 Excel 的功能，它能和个人宏工作簿一样随 Excel 软件自动启动。加载宏中的 Sub 过程可以随时在任意工作簿中调用，而不仅仅在加载宏文件中调用。

制作加载宏的技术将在本书第 14 章详细介绍。

2.6.3 加载项

加载项和加载宏的存在目的相同，只是文件格式不同，设计方式不同。加载项文件通常为 DLL 格式，用 VB、VB.net 或者 Delphi 等软件开发，相对于 xla 和 xlam 格式的加载宏有较强的安全性，不过制作方法更复杂。

2.7 课后思考

1．在 VBE 界面中哪些部件用于存放代码？不同的代码窗口各适合存放什么代码？

2．假设不小心关闭了工程资源管理器和属性窗口（见图 2-37），从而无法查看代码和属性，

用什么办法重新显示工程资源管理器和属性窗口?

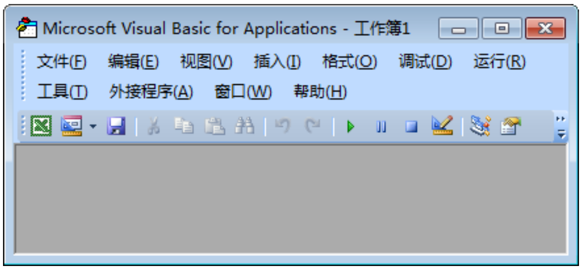


图 2–37　关闭工程资源管理器和属性窗口后的状态

3．用什么办法既可以阻止宏病毒运行，又能让自己编写的代码可以正常执行?

4．保存有宏代码的工作簿适宜采用什么格式?

5．录制一个打开 C 盘“生产表.xlsm”的宏，对宏指定快捷键为【Ctrl+e】，且在工作簿中创建一个按钮，将按钮关联到录制的宏。

第 3 章　从概念开始认识 VBA

VBA 编程涉及较多的理论，在掌握这些基础理论的前提下才可能写出通用性、兼容性好的 VBA 代码。

本章向读者展示 VBA 最基本的概念，从过程的分类开始，到编写过程的语法、递归、参数等，再逐一解析 VBA 中最重要的四大知识体系——对象、属性、方法和事件。

本章要点

- 认识过程
- 关于参数
- 理解对象
- 对象的属性与方法
- 对象的事件

3.1　认识过程

录制宏是一个过程，自定义函数也是过程，所有事件都是过程……认识过程是 VBA 编程的基础。

3.1.1　过程的分类

过程包括三类：Sub 过程、函数过程和属性过程。

Sub 过程的标志是以 Sub 开头，所有录制宏产生的过程都是 Sub 过程。但是带参数的过程、函数过程和所有事件过程都不能通过录制宏产生。

以下过程属于 Sub 过程的标准格式：

```
Sub 提示()                              '放置位置：模块中
 MsgBox "欢迎使用 Excel，今天是" & Date
End Sub
```

在使用 VBA 时，约 90%的情况都是在使用 Sub 过程。

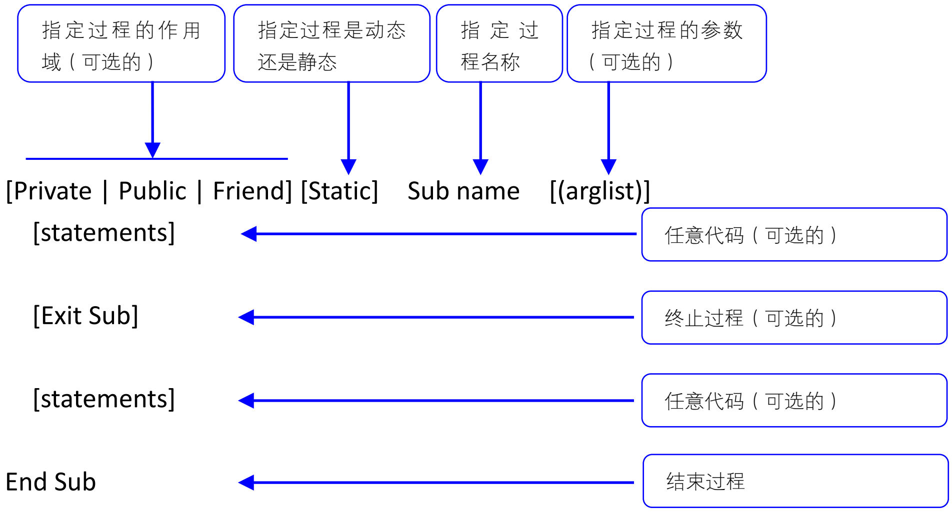
函数过程的标志以 Function 开头，能返回值。以下过程属于函数过程的标准格式：

```
Function 及格率(cell As Range)    '放置位置：模块中
    及格率 = WorksheetFunction.CountIf(cell, ">=60") / WorksheetFunction.CountIf(cell, ">0")
    及格率 = Format(及格率, "0.00%")
End Function
```

属性过程的标志以 Property 开头，在 VBA 中一般不使用属性过程，因此本书不介绍属性过程的编写与应用方法。

3.1.2　Sub 过程的基本语法

Sub 过程的语法如下：



对于以上语法说明需要补充五点。

其一：其中方括号“[]”中的部分表示可选的，可以根据需要决定是否录入。而“|”符号是并列分隔符，表示它前后的内容属于同一级别，只能选其中一项。例如“[Private | Public | Friend] [Static] Sub name [(arglist)] ”实际上相当于以下三句。

[Private] [Static] Sub name [(arglist)]

[Public] [Static] Sub name [(arglist)]

[Friend] [Static] Sub name [(arglist)]

其中 Public 表示所有模块的所有其他过程都可调用这个 Sub 过程；Private 表示只有当前模块中的其他过程才可以访问当前过程；Friend 仅用于类模块，表示当前 Sub 过程在整个工程中都是可见的，但对对象实例的控制者是不可见的。如果这三者都忽略，那么默认当作 Public 处理。

其二：Static 用于声明当前过程是动态过程还是静态过程。有 Static 时表示当前过程是静态过程，其特点是过程中的私有变量的值不会消失；无 Static 时表示当前过程是动态过程，其特点是过程中的私有变量的值将在过程结束后消失。在本书第 5 章关于变量的内容中将会有静态变量与动态变量的应用。

其三：括号“()”中的部分表示过程的参数。例如“(arglist)”，由于参数外边也有方括号，表示参数也是可选的，可以忽略。使用了参数的过程只能通过代码执行，不能通过【Alt+F8】组合键的形式在“宏”对话框中执行此过程。

其四：Sub 过程包含程序外壳和代码两部分，程序外壳包括“Sub Name”和最末一句“End Sub”，这是必选项，任何时候不能省略；代码部分可以省略，不过代码部分才是程序的主体，省略后就不再有意义，所以通常至少会有一句代码。

其五：代码部分“[statements]”表示过程的任意代码，“[Exit Sub]”表示终止过程，该代码允许放在过程中的任意位置，通常配合 If 语句或者 Goto 语句使用，表示在符合某条件时终止过程（查询帮助的关键字：Exit 语句）。

以下代码创建了一个名为“创建工资条”的公有过程，该过程的作用域是所有模块，即任意模块中的其他过程都可以调用此过程。不过省略 Public 也有同样效果，所以通常忽略 Public，只用 Sub 语句指定过程名称即可。

```
Public Sub 创建工资条()

End Sub
```

以下代码创建了一个名为 CopySheet 的私有过程，该过程的作用域是模块内部，即其他模块中的过程不能调用此过程，而且在使用【Alt+F8】组合键打开“宏”对话框时也无法看到此过程，所以不能使用【Alt+F8】组合键来执行此类过程。

```
Private Sub CopySheet()
'代码 1
If Date > #6/1/2016# Then Exit Sub
'代码 2
End Sub
```

在此过程中还使用了“Exit Sub”语句，表示该行代码前面的代码可以随意执行，而它后面的语句则由系统时间决定。如果日期大于 2016 年 6 月 1 日，那么“Exit Sub”语句会终止过程，它后面的代码都不再执行。

以下代码创建了一个名称为 AddSht 的过程，它带有一个名为 ShtName 的参数。所有带有参数的 Sub 过程都不能通过【Alt+F8】组合键调用，而是作为另一个过程的 Sub 过程使用。

```
Sub AddSht(ShtName As String)    '放置位置：模块中
 Worksheets.Add(Sheets(1)).Name = ShtName
End Sub
```

例如以上过程的含义是使用 Worksheets.Add 在第一个表之前创建一个新工作表，并且此表的名称由过程的参数决定，即变量 ShtName。

假设通过另一个过程 Test 来调用过程 AddSht 过程从而新建工作表，可采用以下步骤实现。

（1）在 VBE 窗口中选择“插入”→“模块”。

（2）在模块中录入过程 AddSht 的代码。

（3）继续录入一个新过程的代码。

```
Sub Test()   '放置位置：模块中
'调用过程 AddSht，并对过程的参数赋值为“总表”，表示创建一个“总表”，保存在所有表之前
AddSht "总表"
End Sub
```

（4）单击过程“Test”中的任意位置，使过程“Test”成为当前过程，然后按下【F5】键执行此过程，执行结果是在当前工作簿的第一个表之前新建一个名为“总表”的工作表。

通常是在某个操作需要反复执行多次，或者在多个过程中需要执行某个相同的操作时，需要创建一个带有参数的过程，然后作为其他过程的 Sub 过程执行命令。例如在 10 个过程中都需要新建工作表，然后对工作表命名，那么在过程中直接调用 AddSht 即可，比在每个过程中都编写创建工作表，且对工作表命名更简单。更多关于参数的详情请参阅本章第 2 节。

本例案例文件请参考：..\ 第 3 章\ 3-1 用一个过程调用另一个带参数的过程.xlsm

3.1.3 Sub 过程的命名要求

对过程命名须遵循以下规则。

（1）首字母不能是数字。

（2）不能含有标点符号及空格。

（3）不能使用系统内置的保留字。

（4）在同一个模块中不能与其他过程同名。

（5）长度不能超过 255。

以下是不合格的过程名称。

2008：首字母使用了数字，若用“北京 2008”则没有问题。

AS：系统的保留字，不可用于过程名称。为避免与系统冲突，禁止使用此类名称。

Date：系统的保留字，不可用于过程名称。为避免与系统冲突，禁止使用此类名称。

中国-北京：使用了标点符号之类的特殊符号。

VBA：这个名称虽然可以作为过程的名称，但是建议不用，它会导致通过“VBA.”调用 VBA 的函数和常数时失败。

Range：这个名称虽然也可以使用，但是建议不用，在 VBA 中 Range 表示单元格，为了避免误导阅读代码者，尽量改用与 VBA 中的对象、方法、属性名称不同的名称。

另外，如果在同一个模块中存在多个过程同名时，那么将无法执行代码，它将导致代码编译错误。图 3-1 是在执行过程时由于多个过程同名而产生的错误提示。

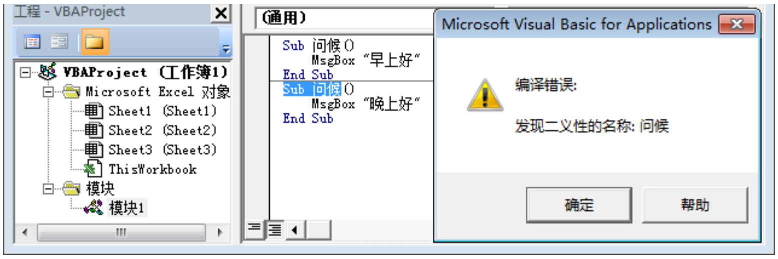


图 3-1　同模块中出现同名过程时的提示

所谓的“二义性的名称”是指一个名称出现了多次，系统无法判断该执行哪一个。所以当看到此类提示时需要马上检查模块中还有哪些过程与当前过程同名。

3.1.4　Sub 过程的调用方法与访问限制

Sub 过程之所以被称之为 Sub 过程是因为它可以被另一个过程调用，即一个过程作为另一个过程的 Sub 过程。虽然函数过程也可以被另一个函数过程或者 Sub 过程调用，不过微软将 Sub 过程的定义限制在 Sub 开头的过程中了。

调用 Sub 过程时受 Sub 过程的声明所限，如果使用 Private 将过程声明为私有的过程，那么该过程只能被当前模块中的过程所调用；如果使用 Public 将过程声明为公有，那么在任意模块中都能调用该 Sub 过程，可以通过以下步骤验证。

（1）插入 3 个模块，且让 3 个模块都使用默认的名称，即模块 1、模块 2 和模块 3。

（2）在模块 1 中录入过程 a，在模块 2 中录入过程 b，在模块 3 中录入过程 c 和 d，4 个过程的代码如下。

```
Public Sub a()
 MsgBox "A"
End Sub
Private Sub b()
 MsgBox "B"
End Sub
Sub c()
 Call a     '调用过程 a
End Sub
Sub d()
```

```
    Call b     '调用过程 b
End Sub
```

图 3-2 呈现了插入的 3 个模块以及模块 1 中的代码。

（3）进入模块 3，选择过程 c，然后按下【F5】键或者单击工具栏中的运行按钮（ ），Excel 会弹出一个对话框，在对话框中将显示信息“A”，表示调用过程 a 成功；

（4）选择过程 d，然后按【F5】键执行过程，将弹出如图 3-3 所示的错误提示框，表示找不到过程 b。

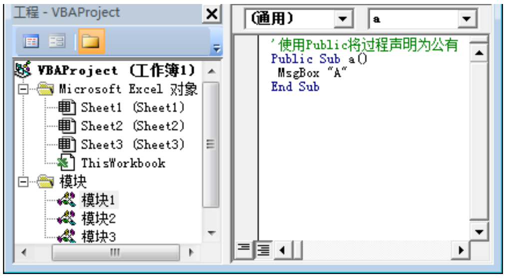


图 3-2 在三个模块中存放 4 个过程

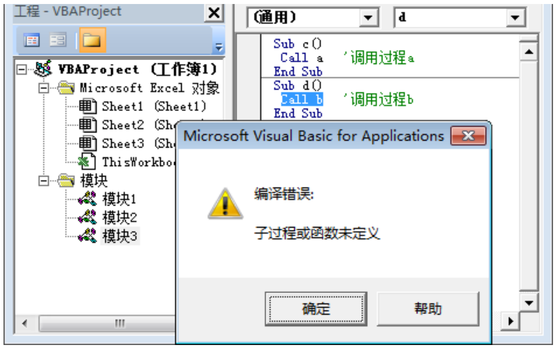


图 3-3 调用私有过程时出错

出现以上现象是因为在模块 2 中的过程使用了 Private 将过程转成私有过程，即模块级过程，该过程只能被同模块中的过程调用。而模块 1 中的过程 a 使用了 Public 将过程声明为工程级过程，在工程中的任意模块中都可以随意调用。

事实上，模块 3 中的两个过程也是工程级过程，即 Public 可以忽略不写。

注意：Call 语句表示调用 Sub 过程，如果 Sub 过程有必选参数，那么在调用时需要将参数一并罗列出来。Call 语句允许省略，所以调用过程 a 也可以简写为一个字母 a 即可。

另外，当使用 Private 将过程声明为私有时，不能在“宏”对话框中看到该过程名称。

本例案例文件请参考：..\ 第 3 章\3-2 工程级与模块级过程.xlsm

3.1.5 过程的执行顺序

当一个过程中有多句代码时，代码总是按从上到下的顺序逐句执行。读者可以使用以下简单的语句进行测试。

```
Sub 问候()        '放置位置：模块中
    MsgBox "早上好"
    MsgBox "中午好"
    MsgBox "下午好"
    MsgBox "晚上好"
End Sub
```

VBA 允许将多句代码写在同一行中，中间用冒号分隔开即可。当多句代码出现在同一行时则按从左到右的顺序执行。可以通过以下代码验证 VBA 代码的执行顺序。

```
Sub 问候 2()       '放置位置：模块中
    MsgBox "早上好":      MsgBox "中午好"
    MsgBox "下午好":      MsgBox "晚上好"
```

```
End Sub
```

也可以随时通过“Exit Sub”终止程序，从而使其后面所有的代码都不再执行。

例如在以下代码中使用了两次 Msgbox 函数，表示会弹出两个信息提示框。不过由于在中间插入了代码“Exit Sub”，所以第二个 Msgbox 语句并不会执行，在“Exit Sub”语句处已经结束了过程。

```
Sub 提示()        '放置位置：模块中
    MsgBox "Excel 很不错!"
    Exit Sub      '此代码表示退出 Sub 过程，不再执行后面的代码
    MsgBox "VBA 也很犀利啊!"
End Sub
```

如果中途调用了另一个 Sub 过程，那么会在执行完该过程中的所有代码后再执行 Call 语句后面的代码。在过程“问候 3”中调用了 Sub 过程“问候 4”，所以在执行完“问候 4”的两句代码后才会执行“问候 3”的最后一句代码。

```
Sub 问候 3()      '放置位置：模块中
    MsgBox "早上好"
    Call 问候 4
    MsgBox "晚上好"
End Sub
Sub 问候 4()      '放置位置：模块中
    MsgBox "中午好"
    MsgBox "下午好"
End Sub
```

当然，VBA 也提供了改变执行顺序的办法，从而适应工作中的实际需求——GoTo 语句。Goto 语句的语法如下。

```
GoTo line
```

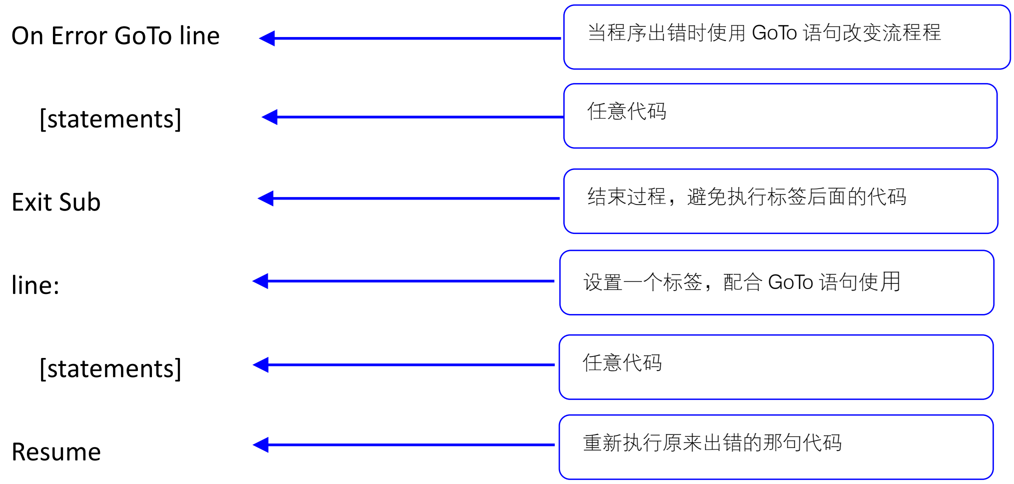
其中参数 line 是标签名称。标签就相当于为代码做一个标记，当 Goto 语句指向此标签时可以改变代码的执行流程，跳转到标签所在位置执行其后的代码。

定义标签的方法是在代码之前插入标签名称，以冒号结尾。以下过程“问候 5”使用标签 A，代码“If Hour(Now) > 20 Then GoTo A”表示如果当前时间大于 20 点，那么执行标签“A”后面的代码，GoTo 语句到标签 A 之间的所有代码都将被忽略。

```
Sub 问候 5()      '放置位置：模块中
    MsgBox "早上好"
    If Hour(Now) > 20 Then GoTo A '如果时间大于 20 点，则执行标签“A”后面的代码
    MsgBox "中午好"
    MsgBox "下午好"
A:                                '设置一个标签：A
    MsgBox "晚上好"
End Sub
```

使用 GoTo 语句可以随意更改代码的执行语句，通常配合 If 语句使用，表示符合某条件时执行某语句，符合另一条件时执行另一语句。

此外 Resume 也能改变代码执行顺序。Resume 表示返回原来的行，它需要配合 On Error GoTo 语句使用。具体语法如下（查询帮助的关键字：On Error 语句）。



其中“line”表示设置的标签，“statements”表示若干句代码，是可选的。其整体含义为当发生错误时，执行标签之后的代码，当执行到“Resume”语句处时再返回到前面出错的语句处继续执行。如果没有错误，那么按从上到下的顺序执行代码，到“Exit Sub”语句处结束过程。

以下代码表示将 Sheet1 工作表中的 A1 单元格复制到总表中，由于担心工作簿中没有总表或者总表被不经意删除导致复制命令出错，所以将 On Error GoTo 语句配合 Resume 使用，从而使代码具有防错功能，当不存在总表时跳到标签“新建”处，创建好总表后再返回原来的地方继续执行。

```
Sub 复制 Sheet1 的 A1 到总表 2()          '放置位置：模块中
    On Error GoTo 新建                  '如果有错误，那么执行标签"新建"之后的代码
Sheets("sheet1").Range("A1").Copy      '复制 sheet1 工作的 A1 单元格
'仅将它的值粘贴到总表的 A1（选择性粘贴：值)
    Sheets("总表").Range("A1").PasteSpecial Paste:=xlPasteValues
    Exit Sub                           '退出程序，避免执行标签之后的错误处理程序。
新建:                                   '设置一个标签，当执行前面的代码出错时就跳转到此处
    Sheets.Add.Name = "总表"            '新建一个空表，且将它命名为总表
    Resume                             '返回到发生错误的语句继续执行（因为已经消除了错误)
End Sub
```

可以先选中代码，然后按【F8】键逐句执行以上代码，看看代码的执行顺序，以及执行代码的结果。可以在工作簿中有总表和无总表两种环境下测试，从而加深对 Resume 的认识。

过程中的 Copy 表示复制单元格，可通过关键字“Range.Copy”在帮助中查询它的具体语法及示例。

本例在于展示 Resume 如何改变程序的执行顺序，对于新建工作表、复制单元格、选择性粘贴等方法的语法和应用将在后面的章节有详细介绍。

本例案例文件请参考：..\ 第 3 章\ 3-3 代码的执行顺序.xlsm

3.1.6 过程的递归

所谓递归是指程序调用自身的一种方法，它是一把双刃剑，合理利用递归能使工作变得更简单，解决某些非递归不能解决的问题。但是如果不小心进入了递归，则可能造成程序崩溃或者溢

出堆栈空间。

以下代码是简单的递归过程，它本身只有一句代码，作用是对 A1 单元格的值加 1。不过由于使用了“Call 累加”，从而使过程可以反复执行，直到溢出堆栈空间时止（VBA 为了保护程序正常运行，避免崩溃，当调用过程次数过多时自动终止，并弹出错误提示框）。

```
Sub 累加()                                    '放置位置：模块中
    Range("A1") .Value= Range("A1") .Value + 1   '将 A1 的值累加 1
    Call 累加                                  '调用过程“累加”，即递归用法
End Sub
```

可以按【F8】键逐句执行，从而观察代码的执行效果，加深对递归现象的理解。

以上过程属于因溢出堆栈空间导致程序出错从而终止代码的案例，事实上有时合理利用递归，也能简化代码。

3.2　关于参数

微软将参数的定义为：传递给一个过程的常数、变量或表达式。仅根据这个简单的定义很难理解什么是参数，以及它有何不可取代的优势。本节详细阐述参数的相关知识，帮助读者理解参数的价值和使用方法。

3.2.1　参数的存在价值

参数分为两类，一类是过程名称中的参数，一类是代码中的参数，它们对于程序的价值体现在不同方面。

1．过程名称中的参数

过程名称中的参数写在过程名称后面的括号中，它能扩展过程的功能，使过程更强大、更灵活。例如“Sub 问候(Time_value as Long)”中的 Time_value 是过程名称中的参数。

过程名称中的参数相当于一个软件的自定义选项，有了足够的自定义选项，软件才能适应工作中灵活多变的需求。例如 Excel 提供宏安全性选项，让用户能根据需求在 4 种选项中选择适合自己的选项，再如打印界面允许自定义上下左右的页边距，而非固定使用相同边距。

同理，拥有参数的过程也将使过程更强大、更灵活，例如过程的功能是对一个当前表的已用数据区域求和，它的功能被限制为指定的工作表和指定的区域。如果对过程使用了参数，那么工作表名称和区域地址可以通过参数从主过程中传递到 Sub 过程，从而使操作对象不再固定，可以随意修改，从而使程序更强大、灵活。在 3.2.3 节中将有相关的案例展示。

2．代码中的参数

代码中的参数通常指 Excel 中各种方法的操作对象或操作方式。参数对于 VBA 的方法而言至关重要。例如在复制区域时，复制到什么地方？只复制值还是要复制值和格式？是否需要转置？这些问题都需要通过参数来控制。有了参数，VBA 才能解决工作中的复杂问题。

3.2.2　过程名称中的参数

过程名称中的参数通常起到传递常数或者变量，强化过程的作用。

下面的代码包括 4 个过程，前面 3 个过程分别对应早、中、晚的信息提示，第 4 个过程“问

候”将根据当前时间来决定调用哪一个过程。4 个过程都存放在模块中。

由于过程没有参数，因此需要使用 3 个过程实现 3 个需求。如果有 10 个需求则需要调用 10 个过程。

```
Sub a()   '放置位置：模块中
    MsgBox "早上好"
End Sub
Sub b()
    MsgBox "中午好"
End Sub
Sub c()
    MsgBox "下午好"
End Sub
Sub 问候()                      '分别调用三个不同过程
    If Hour(Now) < 12 Then      '如果小于 12 点
        Call a                  '调用过程 a
    ElseIf Hour(Now) < 13 Then  '如果小于 13 点（即 12 点～13 点之间）
        Call b                  '调用过程 b
    Else                        '否则
        Call c                  '调用过程 c
    End If
End Sub
```

如果过程可以使用参数，那么问题将化繁为简。

```
Sub Hello(Mystr)                '过程 Hello 有一个参数
    MsgBox Mystr                '提示信息的内容即为参数的值
End Sub
Sub 问候()                      '调用有参数的过程
    If Hour(Now) < 12 Then      '如果小于 12 点
        Call Hello("上午好")    '调用过程 Hello，将传递参数“上午好”
    ElseIf Hour(Now) < 13 Then  '如果小于 13 点(即 12 点-13 点之间)
        Call Hello("中午好")    '仍然调用过程 Hello，将传递参数“中午好”
    Else                        '否则
        Call Hello("下午好")    '仍然调用过程 Hello，将传递参数“下午好”
    End If
End Sub
```

对比两种代码编写思路，可以发现使用参数对于编程的价值。事实上参数在工作中还有更复杂的应用，代码越复杂，参数越能体现其优势。

本例案例文件请参考：..\ 第 3 章\ 3-4 调用有参数与无参数的过程.xlsm

3.2.3 参数的赋值方式

VBA 提供两种参数赋值的方式：按位置赋值和按名称赋值，它们各有优势。

1．按位置赋值

当调用过程或者 Excel 内部的各种方法时，如果它们有多个参数，需要在代码中对参数按预设的顺序逐一赋值，如果位置错误将无法获得需要的结果。对于可选参数直接使用逗号占位即可。

以下代码包括两个过程，第一个过程使用了两个参数，过程中的 MsgBox 代码会调用这两个参数，将它们显示在信息提示框中。

第二个过程调用了三次第一个过程，第一次采用 Call 语句按正常顺序调用，其执行结果如图 3-4 所示；在第二次调用时将 Hello 过程的两个参数位置调换，其执行结果如图 3-5 所示；在第三次调用 Hello 过程时未使用 Call 语句，因此它的参数不需要括号，其执行结果与第一次使用 Call 语句调用时的结果一致。

```
'过程 Hello 有两个参数，对应于信息提示框的内容和标题
Sub Hello(Info, Title)                    '放置位置：模块中
    MsgBox Info, vbOKOnly, Title          '提示信息的内容即为参数的值
End Sub
Sub test() '演示参数的多种赋值方式（放置位置：模块中）
    '使用 Call 调用过程，那么 Hello 过程是 Call 语句的参数，所以 Hello 过程的参数前后需要添加括号，对
'参数赋值时必须与 Hello 过程中的参数顺序一致
    Call Hello("VBA 好犀利", "友情提示")
    '如果将参数的位置顺序弄错了，那么将得到错误结果
    Call Hello("友情提示", "VBA 好犀利")
    '如果不使用 Call,那么 Hello 的参数，不使用括号（执行结果与第一句一致）
    Hello "VBA 好犀利", "友情提示"
End Sub
```

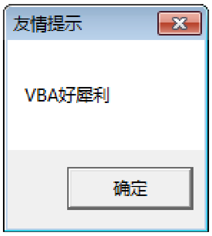


图 3-4　参数顺序正常时的执行结果

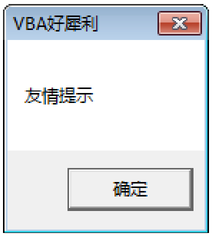


图 3-5　参数顺序调换后的执行结果

2．按名称赋值

按参数的名称赋值时只需要指定正确的参数名称即可，没有顺序要求。赋值的格式如下：

参数名称:=赋值

以下案例中第二个过程 test 调用过程 Hello 时使用了参数名称赋值，因此使用 Call 语句调用过程时参数位置巅倒也不影响结果。而对于未使用 Call 语句调用过程的方法，它的特点是不需要对参数使用括号，参数顺序也同样不影响计算结果。本例中三次调用过程的结果一致。

```
Sub Hello(Info, Title)                 '过程 Hello 有两个参数，对应信息提示框的内容和标题
    MsgBox Info, vbOKOnly, Title       '提示信息的内容即为参数的值
End Sub
Sub test()                             '参数的多种赋值方式
   '如果使用了 Call,那么 Hello 是 Call 的参数，Hello 的参数需要加括号
    Call Hello(Info:="VBA 好犀利", Title:="友情提示")
    '由于使用了按参数名称赋值，所以顺序不影响计算结果
    Call Hello(Title:="友情提示", Info:="VBA 好犀利")
    '如果不使用 Call，那么 Hello  的参数不使用括号（执行结果与第一句一致）
     Hello Info:="VBA 好犀利", Title:="友情提示"
End Sub
```

本例案例文件请参考：..\ 第 3 章\ 3-5 参数的赋值方式.xlsm

3.2.4 可选参数与必选参数

参数分为可选参数与必选参数，必选参数是必须赋值才能让过程正确执行的参数，可选参数则既可以对参数赋值也可以不赋值，当未手动为可选参数赋值时，VBA 会调用该参数的默认值。

前面已说过 Sub 过程的语法如下。

```
[Private | Public | Friend] [Static] Sub name [(arglist)]
```

其中“arglist”是 Sub 过程的参数，由于“arglist”的用法很复杂，所以本小节对它详细剖析。

“arglist”参数的语法如下。

```
[Optional] [ByVal | ByRef] [ParamArray] varname[( )] [As type] [= defaultvalue]
```

从语法可以看出，“arglist”参数也带有很多参数，且所有参数都是可选的。具体如表 3-1 所示。

表 3-1　Sub过程的arglist 参数的参数列表

参　　数	说　　明
Optional	如果使用了该选项，则后续所有参数都必须是可选的，而且必须都使用 Optional关键字声明。如果使用了ParamArray，则任何参数都不能使用 Optional
ByVal	表示该参数按值传递
ByRef	表示该参数按地址传递。默认状态下都按地址传递
ParamArray	用于限制最后一个参数，指明最后这个参数是一个Variant类型的可选的数组。使用 ParamArray关键字后可以提供任意数量的参数。不过它不能与ByVal、ByRef、Optional 一起使用
varname	必选参数。代表参数的变量的名称；遵循标准的变量命名约定
type	表示参数的数据类型，可以是Byte、Boolean、Integer、Long、Currency、Single、Double、Date、String、Object 或 Variant
defaultvalue	当过程的参数使用Optional进行限制时，需要为它指定一个默认值，而defaultvalue正是参数的默认值。如果参数的类型为Object，那么其默认值只能是Nothing

从表 3-1 中可以挑出几个比较重要的信息。

（1）当需要将某个参数设置为可选参数时，在该参数名称前添加 Optional 关键字即可。同时还需要对其赋予默认值，从而当调用过程未指定参数值时 VBA 会调用此默认值。

（2）被 Optional 限制的参数必须是靠后的位置，即它的右边没有必选参数。

（3）Optional 一次只能将一个参数转换成可选参数，而使用 ParamArray 可以将一个参数转换成不确定个数的可选参数。在 Excel 2003 中，过程的参数上限是 30 个，Excel 2007 及以上版本的参数上限是 255 个，所以如果当前只有一个参数，那么 ParamArray 能将它转换成 30 或者 255 个可选参数，类似于 Sum 函数的不确定个数的可选参数。如果在 ParamArray 限制的参数之前还有其他参数，那么应该用 255 或者 30 减去其他参数的个数，得到 ParamArray 可产生的参数个数。

（4）使用 ParamArray 转换参数时，该参数必须使用变体型的数组，例如“ParamArray a()”。

（5）ParamArray 和 Optional 是相互冲突的，使用了其中一个就不能使用另一个。

在编写自定义函数时更需要可选参数，本小节仅通过 Sub 过程来展示可选参数，重点在于帮助读者理解可选参数的设计思路与功能。

在以下过程中通过 Optional 将两个参数转换成可选参数，当调用此过程时，如果不对参数赋值，那么第一参数默认使用“上午好”，第二参数默认使用“友情提示”。

```
'过程 Hello 有两个可选参数
Sub Hello(Optional Info = "上午好", Optional Title = "友情提示") '放置位置：模块中
    MsgBox Info, vbOKOnly, Title   '提示信息的内容即为参数的值
```

```
    End Sub
```

以下过程根据当前时间调用过程 Hello，如果时间小于 12 点，那么会提示“上午好”，提示框的标题是“友情提示”，由于刚好与两个参数的默认值对应，所以直接调用过程名称即可，不需要对参数赋值；当时间小于 13 点时，由于第一参数的值不能再调用默认值，所以只忽略了第二参数；第三次调用过程 Hello 时，两个参数都不再采用默认值，所以在括号中完整地对两个参数赋值。图 3-6、图 3-7 和图 3-8 分别是三次调用过程的执行结果。

```
Sub Test()                                  '放置位置：模块中
    If Hour(Now) < 12 Then                  '如果小于 12 点
        Call Hello                          '直接调用过程 Hello，不需要对参数赋值
    ElseIf Hour(Now) < 13 Then              '如果小于 13 点(即 12～13 点之间)
        Call Hello("中午好")                '第一参数手动指定，第二参数使用默认值
    Else                                    '否则
        Call Hello("下午好", "人事系统提醒您")  '两个参数都手动指定
    End If
End Sub
```

如果 Hello 过程的第一参数不使用 Optional，那么运行过程“Test”时将出现如图 3-9 所示的提示，表示未对必选参数赋值，因此无法继续执行命令。

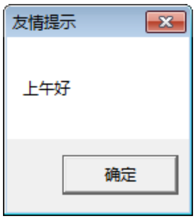


图 3-6　提示 1

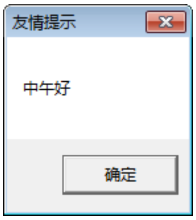


图 3-7　提示 2

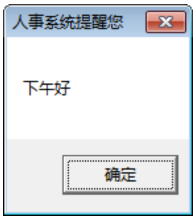


图 3-8　提示 3

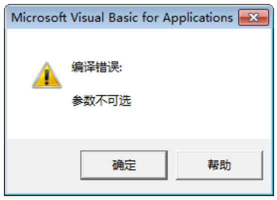


图 3-9　未对必选参数赋值

如果过程的参数未使用 Optional，那么也不可以对它赋予默认值。另外过程“Hello”若需要将其中一个可选参数转换成必选参数的话，只能将第一个可选参数调整为必选参数，VBA 不允许可选参数排在必选参数之前。

本例案例文件请参考：..\ 第 3 章\ 3-6 可选参数与必选参数.xlsm

3.2.5　代码中的参数

代码中的参数通常指 VBA 的方法或者函数的参数，这是系统预定义的参数，而上一小节所述过程参数则由编写过程代码者所定义。

例如 VBA 中的 Range.Copy 方法就有一个可选的参数，其语法如下：

```
Range.Copy([Destination])
```

Destination 参数表示目标区域，当不使用目标参数时，表示将一个 Range 对象（即单元格或者区域）复制到剪贴板中；当使用目标参数时，表示将复制的对象粘贴到目标区域中。可见拥有可选参数能使 Range.Copy 方法更灵活，有更多的可选空间。

```
Sub 复制()                           '放置位置：模块中
    Range("a1").Copy                 '复制 A1 单元格的值到剪贴板
    Range("a2").Copy Range("B2")     '将 A2 的值复制到 B2 单元格中
End Sub
```

在以上过程中，“Range("B2")”是 Copy 方法的参数。由于是可选参数，允许省略。

再如，Range.Insert 方法的语法如下：

```
Range.Insert([Shift],[ CopyOrigin])
```

它有两个参数，参数的功能与参数的值如表 3-2 所示。

表 3-2　Range.Insert方法的参数列表

参数名称	功　　能	可 选 项
Shift	指定单元格的调整方式。例如插入新单元格时原单元格是向下还是向右	xlShiftToRight或者xlToRight表示右移 xlShiftDown或者xlDown表示下移
CopyOrigin	复制格式的对象。即插入的单元格应用左边/上边单元格还是右边/下边单元格的格式	xlFormatFromLeftOrAbove表示左侧或者上方 xlFormatFromRightOrBelow表示右侧或者下方

表 3-2 表明 Range.Insert 方法第一参数控制着原单元格的移动方向，第二参数控制着新单元格的格式应用对象。可以通过以下代码验证表 3-2 的内容。

```
Sub 在 B2 前插入新单元格 1()  '放置位置：模块中
    '在 B2 单元格处插入一个单元格，原单元格下移，新单元格采用上方单元格的格式
    Range("B2").Insert Shift:=xlDown, CopyOrigin:=xlFormatFromLeftOrAbove
End Sub
Sub 在 B2 前插入新单元格 2()  '放置位置：模块中
    '在 B2 单元格处插入一个单元格，原单元格下移，新单元格采用下方单元格的格式
    Range("B2").Insert Shift:=xlDown, CopyOrigin:=xlFormatFromRightOrBelow
End Sub
```

可以对 B1 和 B2 单元格设置不同格式，然后分别执行两个过程，从而比较两者的差异。

此外，由于参数的赋值方式有两种，所以以上过程也可以改为对参数按顺序赋值，代码如下：

```
Sub 在 B2 前插入新单元格 3()  '放置位置：模块中
    '在 B2 单元格处插入一个单元格，原单元格下移，新单元格采用下方单元格的格式
    Range("B2").Insert xlDown, xlFormatFromRightOrBelow
End Sub
```

读者不必担心忘记两个参数的顺序，从而导致在赋值时出错，因为 VBA 会自动提示每个参数的顺序。当录入对象的方法名称后再加一个空格，VBA 就会自动在其下方罗列出参数名称，其中字体加粗者表示当前需要录入的参数，如图 3-10 所示。

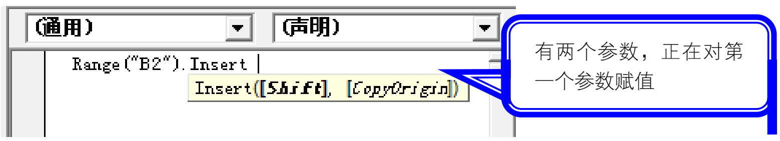


图 3–10　参数的提示

本例案例文件请参考：..\ 第 3 章\ 3-7 Range.Copy 与 Range.Insert 方法.xlsm

3.3　理解对象

VBA 是面向对象的程序语言，VBA 操作的主体就是各种对象，理解什么是对象及如何用代码来表示对象，这对 VBA 而言极其重要。

3.3.1　什么是对象

Excel 对对象的定义是代码和数据的组合，可将它看作单元，例如控件、窗体或应用程序部件。很显然，从这个定义中很难明白什么是对象。

通俗来说 VBA 的对象是指被 VBA 代码执行操作的目标，这些目标绝大多数都可以用肉眼看得见。例如工作表、单元格、字体、透视表、图形、菜单等。Excel 中所有能看到的元素都是对象。可见对象离我们并不遥远，我们每天都要接触大量的对象。

VBA 用名词来表示对象，用动词来表示对象的方法，例如“关闭所有工作簿”的代码如下：

```
Workbooks.Close
```

其中关闭(Close)是动词，工作簿对象(Workbooks)是名词，这里的工作簿对象(Workbooks)就是 VBA 的对象，而关闭（Close）则是操作对象的方法。

“在第 2 行之前插入新行”的代码如下：

```
Range("2:2").Insert
```

其中 Insert 是动词，第 2 行 Range("2:2")是名词，这里的 Range("2:2")就是对象。

“创建一个图表”的代码如下：

```
Charts.Add
```

其中创建（Add）是动词，图表（Charts）是名词，这里的 Charts 就是对象，表示第 2 行。

每个对象都是 Excel 的组成元素之一。

Excel 的常见对象如表 3-3 所示。

表 3-3　常见对象的写法与说明

对　　象	对象名称	说　　明
应用程序对象	Application	即Excel程序，属于Excel中的最高级别对象
工作簿对象	Workbook	每一个文件都属于一个工作簿，允许同时存在多个工作簿
工作表对象	Worksheet	工作簿的子对象，包含Sheet1、Sheet2、Sheet3等工作表对象
单元格对象	Range	工作表对象的子对象，如A1、F10、B888等都是单元格
图形对象	Shape	图形对象包括图表、图片、艺术字、文本框等对象
图表对象	Chart	包括图表工作表和嵌入式图表

其中工作簿对象、工作表对象、单元格对象、图形对象和图表对象都支持复数形式，从而表示对象集合。例如代码“Workbooks.Close”中的 Workbooks 就表示工作簿集合，即所有工作簿。表 3-4 中罗列了几个常见的对象集合及其使用说明。

表 3-4　常见的对象合集

对　　象	对象名称	说　　明
工作簿对象集合	Workbooks	表示所有工作簿，代码“Workbooks.Close”可关闭所有工作簿
工作表对象集合	Worksheets	表示所有工作表，代码“Worksheets.PrintOut”可打印所有工作表
单元格对象集合	Cells	表示所有单元格，代码“Cells.Clear”可清除所有单元格的值
图形对象集合	Shapes	表示所有图形对象，代码“ActiveSheet.Shapes.SelectAll”可选择所有图形对象
图表对象集合	Charts	表示所有图表对象，代码“Charts.Delete”可删除所有图表对象

3.3.2　对象的引用层次

Excel 的对象具有层级结构，多数对象有一个或者多个子对象。

Excel 的最顶层对象是 Excel 应用程序，用 Application 表示，其他所有对象都是它的子对象或者子对象的子对象。

Application 对象的直接子对象有工作簿对象 Workbook、窗口对象 Window、名称对象 Names（专指应用程序的名称）、活动工作表 ActiveSheet（非活动工作表是工作簿的子对象，而不是 Application 的直接子对象）、活动单元格 Activecell（非活动单元格是工作表的直接子对象）等数十个。

在书写对象与子对象时采用“对象名称.子对象名称”的方式，例如：

（1）Application.Workbooks ("生产表.xls")——表示引用 Excel 对象的名为“生产表.xls”的工作簿子对象

（2）Application.AddIns(2)——表示引用第 2 个加载宏对象，它是 Application 的子对象

事实上，绝大多数时候都可以省略 Application 对象，因为顶层对象只有一个，省略后也不会造成混乱。如果多个对象都有同名的子对象，则需要书写完整。例如：

```
Workbooks("财务表.xlsm").Worksheets(2).Range("A1")
Workbooks("工作簿 1").Worksheets(2).Range("A1")
```

第一句代码表示引用“财务表.xlsm”的第 2 个工作表中的 A1 单元格，第二句代码表示引用未保存的工作簿“工作簿 1”的第 2 个工作表的 A1 单元格（工作簿名称中无后缀名就表示此工作簿暂未保存）。

由于所有工作簿都可能出现“第 2 个工作表的 A1 单元格”这种子对象，所以有必要将每一层对象的名称书写完整，不能省略工作簿对象，直接引用其子对象，例如工作表或者单元格对象。

如果不写工作簿名称，直接调用工作表，那么 VBA 将视为用户在调用活动工作簿中的工作表，从而与实际需求相悖。

生活中其实也不乏这种实例。例如在拨打 114 查询电话时，由于每个地区都有 114 服务，那么为了区分就要求在 114 前加拨区号。例如查成都的某个电话需要拨打 028114，查南昌的某个电话需要拨打 0791114。如果需要调用本地的 114 服务台，则可以省略本地区号，直接拨打 114 即可。

在 VBA 中调用活动对象时，包括活动工作簿、活动工作表、活动单元格等，可以忽略其父对象名称，只有操作非活动对象的子对象时，才需要在代码中将父对象与子对象的名称一并书写完整。

对于对象的详细引用方式请参考本书 4.2 节。

3.4 对象的属性与方法

对象的方法和属性是相当重要的两个概念，几乎每次编写代码都会接触到对象与对象的方法、属性。本节讲述属性与方法的概念，以及如何调用对象的方法与属性。

3.4.1 认识属性与方法

对象可以理解为一个物体，例如一辆车。

而对象的属性是指属于对象的、能看到的也可能看不到的一些特点，例如颜色、大小、重量等。具体到 VBA 中，工作表（即 Worksheet 对象）的属性有哪些呢？位置、类型、是否保护等，就是它的属性。以下三句代码用于获取工作表对象的位置、类型、是否保护三个属性。

```
Sub 获取工作表的属性()                         '放置位置：模块中
MsgBox Worksheets("财务报表").Index  '获取“财务报表”的位置，即从左到右的顺序编号
'获取“财务报表”的类型，-4167 代表工作表，-4109 代表图表
    MsgBox Worksheets("财务报表").Type
     '判断“财务报表”是否保护，值为 False 时表示未保护，值为 True 表示已保护
MsgBox Worksheets("财务报表").ProtectScenarios
End Sub
```

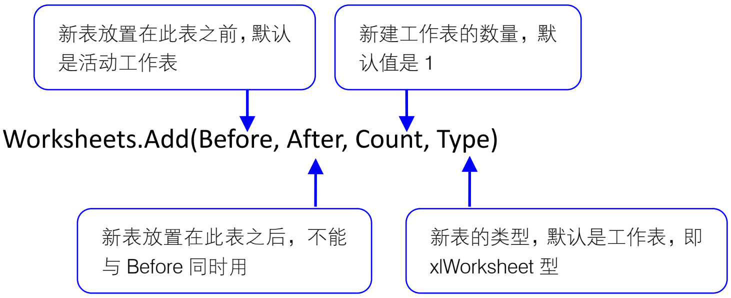
事实上，不仅仅是对象的一些特点算作对象的属性，对象的子对象也属于对象的属性，例如

单元格是工作表的一个属性，工作表中的透视表也是工作表的属性，而工作表又属于工作簿的一个属性。

对象的方法是指对象可以执行的一个动作。对于汽车而言，发动汽车这个动作可算作方法，当然停车也是一个方法。对于 VBA 而言，工作表的创建动作就属于工作表对象的方法，代码如下。

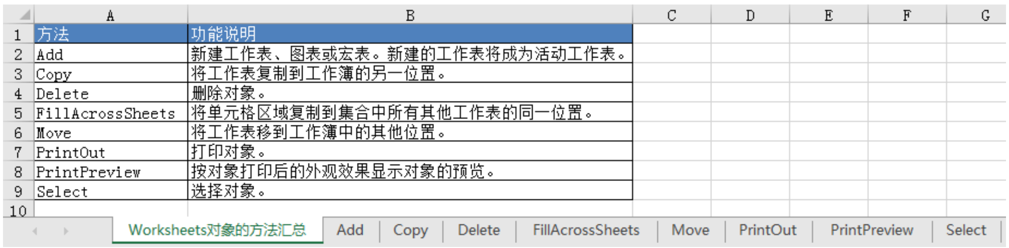
```
Sub 创建工作表()        '放置位置：模块中
  Worksheets.Add        '创建一个工作表，此处 Worksheets 是对象，Add 则是它的方法
End Sub
```

代码中 Add 表示创建一个新工作表，它属于 Worksheets 对象的方法，其完整语法如下。



方法有四个可选参数，第一参数 Before 用于指定一个工作表，创建的新工作表将位于该表之前；第二参数 After 也用于指定一个工作表，创建的新工作表将位于该表之后，它不能与 Before 参数同时使用；第三参数 Count 表示一次性新建的表的个数；第四个参数表示新表的类型。如果省略所有参数，那么 Add 方法将在活动工作表之前创建 1 个工作表。

工作表集合的方法有很多，有什么办法可以找到它的所有方法呢？Excel 2010 的帮助系统很完善，也比较容易查询，从 Excel 2013 开始帮助系统的使用体验差了许多，不便于查询，因此笔者将它整理一份存放在随书案例文件中，文件名称为“Worksheets 对象方法列表.xlsx”。图 3-11 即为 Worksheets 对象方法列表的部分截图。



	A	B
1	方法	功能说明
2	Add	新建工作表、图表或宏表。新建的工作表将成为活动工作表。
3	Copy	将工作表复制到工作簿的另一位置。
4	Delete	删除对象。
5	FillAcrossSheets	将单元格区域复制到集合中所有其他工作表的同一位置。
6	Move	将工作表移到工作薄中的其他位置。
7	PrintOut	打印对象。
8	PrintPreview	按对象打印后的外观效果显示对象的预览。
9	Select	选择对象。
10		

Worksheets对象的方法汇总 | Add | Copy | Delete | FillAcrossSheets | Move | PrintOut | PrintPreview | Select

图 3-11　Worksheets 对象方法列表

如果读者很在意查看帮助的便捷性，那么建议安装 Excel2010，本书的 VBA 知识点在 Excel 2010、Excel 2013 和 Excel 2016 中都是通用的。

本例案例文件请参考：..\ 第 3 章\ 3-8 获取工作表的属性.xlsm

3.4.2　自动调用属性与方法

为了方便用户学习，以及确保录入代码的准确性，VBA 为对象提供了快速信息，快速信息中包含了对象的方法和属性。以 Worksheets 对象为例，自动调用它的属性或者方法只需要在代码

窗口中录入对象名称加一个小圆点即可，如图 3-12 所示。

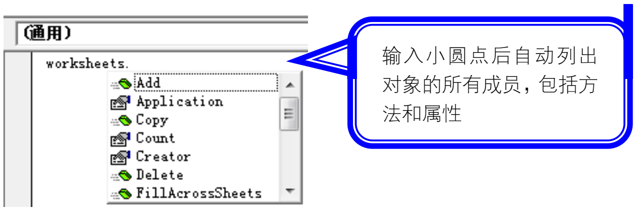


图 3–12 对象的快速信息

在图 3-12 中，下拉列表中的绿色图标表示它是对象的方法，而黑色图标为对象的属性。从图中可以看到工作表对象集合有 Add（创建）、Copy（复制）、Delete（删除）等方法，它们的共同点是都属于动词。

不过此方法并非对所有对象都生效，部分对象不提供快速信息。例如对复数形式的对象使用参数引用其子对象后，该子对象不提供快速信息。例如以下对象都没有快速信息：

```
Sheets ("A")
Sheets ("生产表")
Workbooks ("一月.xls")
Cells(5,5)
```

不过解决办法也比较简单，使用变量替代对象，然后在录入变量时就可以调用对象的属性和方法。在本书第 5 章会讲到变量的用法。

通过快速信息下拉列表录入代码的优点有两个，其一是提升录入代码的准确度，因为对象、属性、方法的名称都是英文字母，有些属性超过 10 个字符，手动录入代码时出错的概率较大，例如将 Worksheetfunction 写成 WorkSheetsFunction，而从下拉列表中选择目标则不可能产生拼写错误的问题；其二是可以避免花费大量的时间去背单词，直接从列表中选择即可，或者只要知道首个字母是什么就差不多能正确地录入整个单词了。

3.4.3　怎样才算完整的 VBA 语句

前面的章节中提到很多代码，读者可能会发现部分代码可以执行成功，部分代码无法执行，例如以下两句都无法正常执行。

```
ActiveSheet.Range("B:B")
Workbooks("生产表.xlsm")
```

这里涉及到一个新的知识点——如何判断一句代码是完整的代码。

对于 VBA 而言，一句完整的代码必定有一个动作，例如赋值、修改某个属性、打开或者关闭对象、声明一个变量或调用一个过程等。

以下两句代码只是对象名称，属于名词，因此不构成一句完整的代码。

```
Workbooks("生产表.xlsm")
Workbooks("Book1")
```

分别执行以上两句代码，将弹出错误提示“属性的使用无效”和“对象不支持该属性或方法”。

而以下代码属于完整的代码：

Workbooks("生产表.xlsm").Close——关闭工作簿，有对象有方法，即为完整代码

MsgBox Workbooks("生产表.xlsm").Name——提示工作簿名称，提示即为动作

Worksheets.Add Sheets("Sheet3")——创建新工作表，创建即为动作

Range("A1")＝123——对单元格 A1 赋值，等号表示赋值，也算一个动作

以下代码的写法是错的，虽然有对象、有方法，但是并不构成一句完整的代码：

```
Worksheets.Add(Sheets("Sheet3"))
```

当对象的方法带有参数时，如果在该行代码中除对象名称、对象的方法以及方法的参数外没有别的代码，那么在对象方法的参数前面不能使用括号，正确的代码如下：

```
Worksheets.Add Sheets("Sheet3")
```

如果在该行代码中除了对象名称、对象的方法以及方法的参数外还有别的代码，那么在方法的参数前后必须添加括号，例如以下三种形式都需要对参数添加括号。

形式一：

```
MsgBox Worksheets.Add(after:=Worksheets(1), Count:=1).Name
```

形式二：

```
Worksheets.Add(after:=Worksheets(1), Count:=1).Name = "总表"
```

形式三：

```
With Worksheets.Add(after:=Worksheets(1), Count:=1)
End With
```

还有一类由多句代码组成的语句，判断这类语句是否完整只需要看是否包括起始和结束语句即可。例如 With 语句：

```
With object
End With
```

With object 和 End With 是一个整体，缺一不可。

以下是多行结构的 If 语句，其中 Else 是可选的：

```
If condition Then
[else]
End If
```

例如过程中忘记了写“End If”，执行过程时将弹出错误提示“块 If 没有 End If”。

3.5 对象的事件

对象除方法、属性外，还要有事件。属性指对象的某些特征，而方法指对这个对象执行某些操作，是一个动词，那么事件是什么呢？它有何价值？如何分类？这是本节的教学重点。

3.5.1 什么是事件

假设汽车是一个对象，那么汽车的大小、颜色等是它的属性，而发动汽车这个动作则属于它的方法。在 Excel 中，新建工作表是一个动作，激活工作表也是一个动作；移动单元格是动作，修改单元格的值也是一个动作。

微软公司根据工作需求对部分动作设置了相应的反应，即在对某个对象执行一个动作后，可以让 Excel 产生若干相应的操作，从而方便工作。例如执行“在单元格中录入产品名称”这个动作时，Excel 自动在其右方的单元格中生成该产品的单价，这就是一个极具价值的功能，而完成此需求就要依赖 Excel 的事件了。

Excel 有很多对象，对每个对象都可以执行很多动作，但是 Excel 只对少部分的对象和少部分动作设计了事件。例如选择单元格有 SelectionChange 事件，改变单元格的值有 Change 事件，右击单元格有 BeforeRightClick 事件，但是修改单元格的背景颜色、字体等操作没有对应的事件。

用好事件可以让工作实现自动化，从而提升工作效率。例如在开启工作簿时自动检查工作表中客户付款时间是否超过预设的期限，或者在开启工作簿时自动执行多工作簿数据合并，以及在

录入产品名称时自动生成对应的单价等。当然也可以在新建工作簿时预设页面打印的边距、页脚等。

简而言之，Excel VBA 的事件就是可被 VBA 识别的某些操作，可以在这些操作发生时调用开发者指定的命令，从而完成工作需求。

3.5.2　事件的存在价值

Excel 事件的本质是 Excel 的对象在执行某些操作时允许开发者执行自定义的操作。

或许用案例来说明事件的功能会更有说服力，也更易于理解：在开启工作簿对象时会自动触发预设的事件，开发者可以在事件中编写命令，执行一切需要的操作，从而实现打开工作簿就自动执行命令。再如新建工作表时，可以调用与工作表相关的或者无关的其他过程，从而实现操作的自动化。事件所关联的过程将在事件发生时自动执行，所以事件也被称为自动化过程，它不需要人工调用。而且事件发生时代码也随之执行，可见对象的事件赋予了对象一些更灵活的特性，能使报表更智能。

可以想象一下事件对于编程的重要性：以往需要单击菜单、调用快捷键或者使用【Alt+F8】组合键执行的过程，配合事件使用时可自动化执行，只要满足条件就会执行相应的代码。

以下代码可以实现开启工作簿时自动在状态栏显示当前系统日期：

```
Private Sub Workbook_Open()     '开启工作簿时自动执行代码（放置位置：ThisWorkbook）
  Application.StatusBar = Date   '在状态栏显示当前系统日期
End Sub
```

这是一个工作簿事件，代码必须放在 ThisWorkbook 对象的事件代码窗口中，当打开此工作簿时会自动执行以上过程，从而在状态栏显示当前日期（见图 3-13）。代码中的“Application.StatusBar”表示 Excel 的状态栏，Date 函数则用于获取当前系统日期。

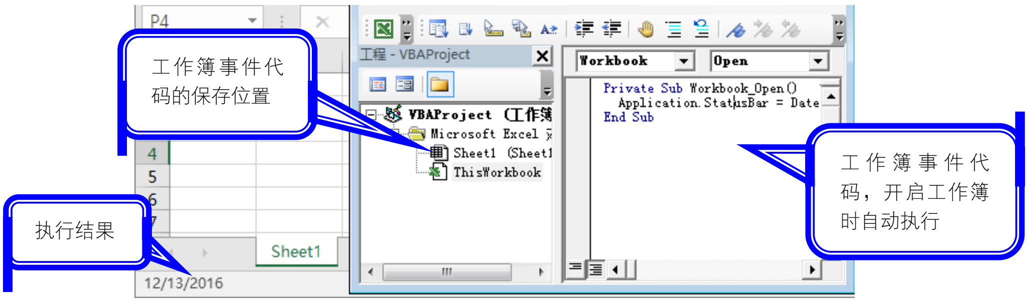


图 3–13　工作簿事件的代码保存位置与执行结果

3.5.3　事件的分类与代码录入方式

Excel 的事件包括 Excel 应用程序事件、工作簿事件、工作表事件、窗体事件和窗体中的控件事件，每个事件都依附在事件的主体对象之上。例如 Sheet1 的事件只属于 Sheet1 工作表，其代码保存在 Sheet1 对象的事件代码窗口中，只有 Sheet1 工作表与事件相关的操作才能触发事件。

事件过程的名称包含两项内容：对象和动作。换言之，事件由对象和动作构成，我们可以通过事件过程的命名看出端倪。

以下是工作簿的 Open 事件，Workbook 是对象，Open 是动作。

```
Private Sub Workbook_Open()

End Sub
```

以下是工作表的 Activate 事件，Worksheet 是对象，Activate 是动作。

```
Private Sub Worksheet_Activate()

End Sub
```

以下是窗体的单击事件，UserForm 是对象，Click 是动作。

```
Private Sub UserForm_Click()

End Sub
```

所以判断一个过程是不是事件，可以通过过程的命名方式来判断，当然对于有参数的事件过程还得确保参数正确以及代码的保存位置正确，任何一个环节有问题事件都不会正常执行。

鉴于事件过程的命名有规律性，因此 VBA 提供了简单的方法来产生过程的名称，从而提升录入代码的速度以及确保命名的准确性。

以录入工作簿的 SheetChange 事件为例，操作步骤如下。

（1）双击工程资源管理器中的“ThisWorkbook”对象，从而打开工作簿事件代码窗口（见图 3-14）。

在代码窗口的上方有两个下拉列表窗口，左边窗口标注了“通用”，那是对象窗口。右边窗口标注了“声明”，那是过程窗口，即事件的动作，两者组合成为事件的名称。



图 3-14　ThisWorkbook 的事件代码窗口

（2）单击对象窗口，从下拉列表中选择“Workbook”，此时代码窗口中将自动出现 Open 事件的代码，这是默认的工作簿事件代码；

（3）单击过程窗口，从下拉列表中选择“SheetChange”，即可产生 Workbook 对应的 SheetChange 事件的代码。对象窗口和过程/动作窗口分别如图 3-15 和图 3-16 所示。



图 3-15　对象窗口

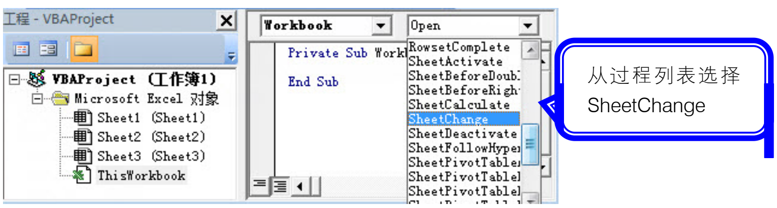


图 3-16　选择过程/动作

在执行完以上步骤后，代码窗口中将自动产生需要的代码：

```
Private Sub Workbook_SheetChange(ByVal Sh As Object, ByVal Target As Range)

End Sub
```

由于在产生 SheetChange 事件的代码之前会产生 Open 事件的代码，所以在产生 SheetChange 事件的代码后手动删除 Open 事件的代码即可。

任何事件都可以通过上述方法录入过程的程序外壳，即过程的命名语句和结束语句。

在工程资源管理器中正确地选择对象是产生正确事件过程代码的前提。

通过 ThisWorkbook 对象只能产生工作簿事件，双击工作表 Sheet2 则只能产生与 Sheet2 相关的工作表事件，在窗体的代码窗口中只能产生窗体或者与窗体中控件相关的事件。

3.5.4 事件的参数

多数事件过程都有参数，这些参数通常与触发事件的条件相关。例如以下工作簿的 SheetActivate 事件表示激活本工作簿中的指定工作表时触发的事件：

```
Private Sub Workbook_SheetActivate(ByVal Sh As Object)

End Sub
```

此事件过程的参数 Sh 代表触发此事件的工作表，该表隶属于代码所在的工作簿。假设工作簿中有 3 工作表，激活其中的 Sheet2 时触发了 Workbook 对象的 SheetActivate 事件，那么参数 Sh 就代表 sheet2。

以下工作表事件表示选区发生变化时触发的事件：

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)

End Sub
```

此事件过程的参数 Target 代表当前被选中的区域（也可能是单个单元格）。选择工作表中的任意区域都会触发 Worksheet 对象的 SelectionChange 事件，但是参数 Target 只代表当前选中的区域，即事件发生时，选中了哪个区域，参数事件的参数 Target 就代表哪个区域。

事件过程的参数类型是不能修改的，事件过程的参数名称则允许修改，但是使用默认的名称最好，从而确保所有事件过程命名保持一致，同时也有利于用户阅读和理解。

事件过程的参数和事件名称一样可以自动产生，而不必手动录入。事件过程的参数不允许删除，且事件过程的参数全是必选参数。

事件过程的参数在工作中相当有用，例如工作表的 SelectionChange 事件的参数 Target 代表当前选中的区域，那么通过以下代码可以在状态栏显示选区的数值合计：

```
'放置位置：Sheet1 的代码窗口中
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
  Application.StatusBar = Target.Address & "合计：" & WorksheetFunction.Sum(Target)
End Sub
```

其中 Address 表示单元格的地址，Target.Address 则表示选中区域的地址。

具体操作步骤如下。

（1）在 Sheet1 工作表中随机录一些数据；

（2）使用【Alt+F11】组合键打开 VBE 窗口；

（3）双击工程资源管理器中的 Sheet1 对象，从而打开 Sheet1 工作表的事件代码窗口；

（4）单击右边的对象窗口，从列表中选择“Worksheet”，此时在代码窗口中将自动产生 Worksheet 对象的 SelectionChange 事件代码。

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)

End Sub
```

这是因为 SelectionChange 事件是 Worksheet 对象的默认事件，所以在选择 Worksheet 对象时会自动产生事件过程的程序外壳，如果是其他事件，那么还需要继续在过程列表中选择过程。

（5）在 SelectionChange 事件的程序外壳中间手动输入以下代码：

```
Application.StatusBar = Target.Address & "合计：" & WorksheetFunction. Sum(Target)
```

最终效果如图 3-17 所示。

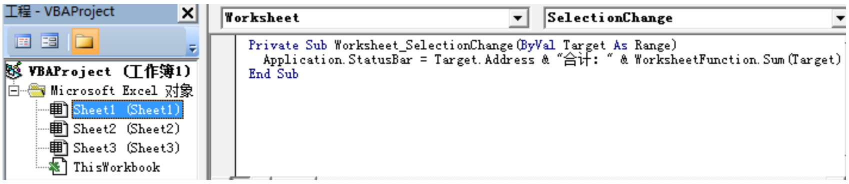


图 3–17　Sheet1 的 SelectionChange 事件代码位置

（6）返回 Sheet1 的工作表界面，任意选择单元格或者区域，在状态栏中将产生选区的地址以及选区数值的合计。事件过程的执行结果如图 3-18 所示。

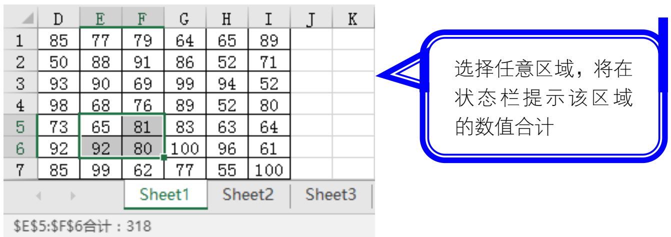


图 3–18　在状态栏显示选区的合计

此部分阐述了思路与基础理论，让读者了解什么是事件，以及事件过程的录入方式和调用方式。在基本功扎实后可以将诸多看似无用的知识点组合成相当有用的功能，在本书的下半部分会有很多案例应用，届时读者可以再深入体会。

关于事件的更详细的说明以及案例演示请参阅本书第 8 章，本章仅对事件的基础概念有所了解即可。

本例案例文件请参考：..\ 第 3 章\ 3-9 在状态栏显示选区的合计.xlsm

3.6　课后思考

1．Sub 过程和自定义函数有什么区别?

2．对参数赋值时有哪些赋值方式? 各自的特点是什么?

3．编写 Sub 过程时，如何为过程创建一个可选参数? 创建可选参数有何规则?

4．在书写某个对象的方法时，如果该方法具有参数，如何判断其参数名称的前后是否需要添加括号?

例如通过 ActiveSheet.Shapes 对象的 AddShape 方法创建新的自选图形，包括矩形、线条、五角星、箭头等图形。它有 5 个参数，例如图形对象的类型、左边距、上边距、宽度和高度，书写它的参数时在什么情况下需要括号，什么情况下不需要括号?

5. 简要描述对象、对象的属性和对象的方法各代表什么。

第 4 章　对象及其层次结构

实际工作中，几乎每段 Excel VBA 的代码都会涉及对象，正确地引用对象和了解对象的层次结构对于学习 VBA 编程尤其重要。

本章重点展示对象的结构和常见对象的引用方式，以及一些简单的应用。

本章要点

- ◆ 查看所有对象
- ◆ 对象的层次
- ◆ Range 对象
- ◆ 图形对象
- ◆ 表对象
- ◆ 工作簿对象
- ◆ Excel 应用程序对象

4.1　查看所有对象

Excel 2010 有 260 多种对象，Excel 2016 有 300 多种对象，将它们全部背下来很难，事实上也完全没有必要花费精力去记这种知识。VBA 提供了关于对象的足够详尽的帮助，读者仅需懂得如何将这些信息调出来即可，这也正是本节的主要内容。

4.1.1　从对象浏览器查看对象

Excel 提供了对象浏览器用于查看所有的对象，包括 Excel 对象及其子对象、VBA 对象、窗体对象、Office 对象、VBE 对象等，同时在窗口右方还显示了当前对象的方法与属性。

调用对象浏览器的快捷键是【F2】，打开对象浏览器后可以在“工程/库”下拉列表中将默认的“所有库”修改为“Excel”，然后在下方的“类”列表中会出现 Excel 的子对象。而选中对象后在右方窗口中会产生当前对象的所有成员，包含属性和方法，在图 4-1 中，由于在“类”列表中选择了单元格对象，其类别名称为 Range，因此在右边成员列表中会显示单元格对象相关的所有属性和方法。

4.1.2　从帮助中调用对象的详细信息

在图 4-1 中显示了 Excel 的所有对象，以及选中对象的属性和方法，事实上还可以调用更详细的帮助信息。如果要查看对象相关的详细信息，那么在选中对象后单击右上角的问号按钮即可，如果要查看某属性或者方法的详细信息，那么在选中该属性或者方法名称后单击右上角的问号按钮。

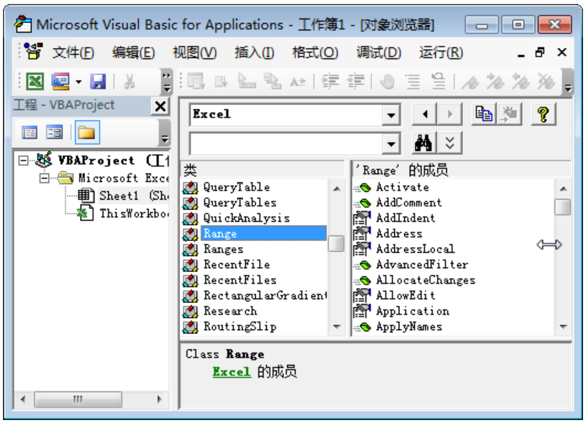


图 4–1 对象浏览器

例如想要查看 Worksheet 对象的详细信息，那么在“类”列表中选择了 Worksheet，然后单击右上角的问号按钮，Excel 会打开以下网址：

https://msdn.microsoft.com/zh-cn/library/office/ff194464(v=office.15).aspx

在该网站中列出了 Worksheet 对象的功能说明，同时提供多个应用案例，效果如图 4-2 所示。

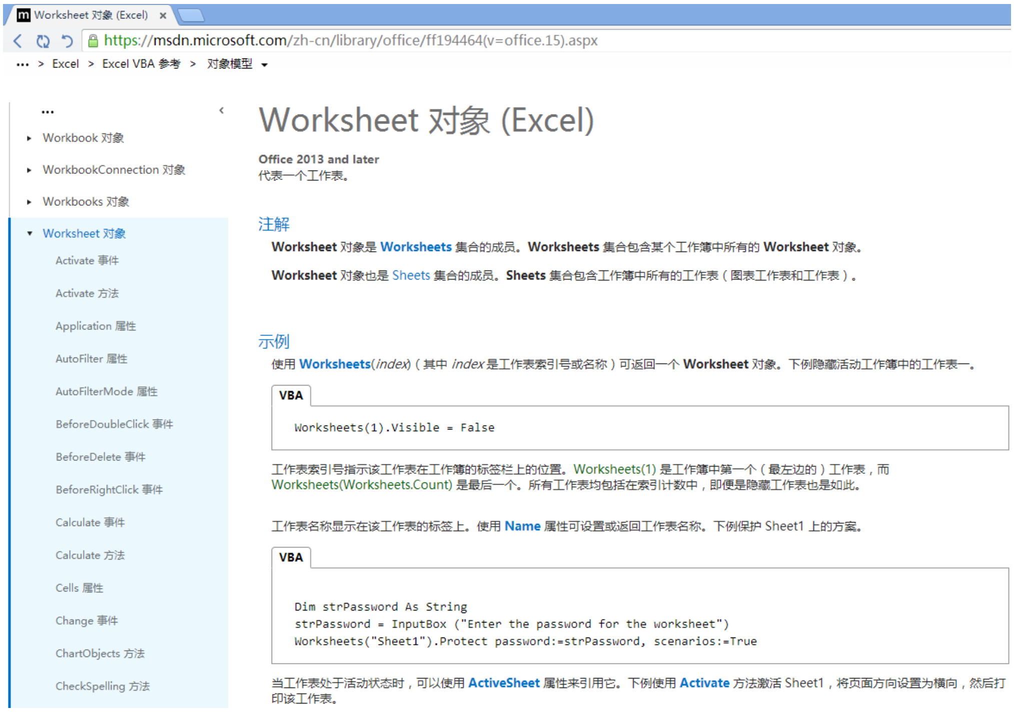


图 4–2 Worksheet 对象的帮助信息

在示例页面的左方还可以看到 Worksheet 对象的所有事件名称、属性名称和方法名称。

注意：尽管 Excel 有 300 多个对象，但常用的对象就 10 多个，包含 WorkBook（工作薄）、Worksheet（工作表）、Range（单元格）、Comment（批注）、WorksheetFunction（工作表函数容器）、Chart（图表）、Sort（排序方式）、AutoFilter（自动筛选）、Error（错误）、FormatCondition（条件格式）、Validation（数据有效性规则）、Name（名称）、Shape（图形）。

4.2　对象的层次与引用方式

VBA 的对象是有层次结构的，不同层次的对象有不同的引用方法。本节针对各种对象的引用方式分类详述，促进读者对对象有更深刻的认识。

4.2.1　对象的层次

对象具有层次结构，就像省、市、县、镇的层级一样。在引用对象时需要层级关系逐层引用，从而表示各对象的关系，同时也能确保引用的正确性。

例如在填写地址时可用“广东省-东莞市-长安镇”，而 VBA 中引用对象时则按“Application.Workbooks("生产表.xlsm").Sheets("Sheet1")”的形式引用，此案例语句表示引用的是 Excel 程序中“生产表.xlsm”工作簿下的“Sheet1”工作表。将前面的地址和后面的 Excel 对象引用进行比较，是否更容易理解呢?

在填写地名时，填写“广东省-广州市”其实和“广州市”是一样的，所有人都知道广州市属于广东省，所以有时我们可以忽略它。但是特殊情况下需要特殊处理，例如贵阳有一个白云区，广州市也有一个白云区，如果省略城市名称就会造成混乱。VBA 中的对象层次结构和它一样，某些对象的父对象是固定的，那么在引用时可以省略其父对象名称，而某些对象则必须完整地表示其父对象才能引用成功。例如：

对于 Workbooks("生产表.xls")而言，任何工作簿都隶属于 Excel 对象 Application，因此忽略其父对象，不会造成混乱。而完整的引用方式应该是 Application.Workbooks("生产表.xls")

对于 Workbooks("生产表.xls").Sheets(3)而言，由于每个工作簿都可能有 Sheets(3)这个工作表对象，因此在引用“生产表.xls”的子对象 Sheets(3)时不能忽略其父对象，否则会造成混乱。

接下来将向读者展示多种引用对象的方式，同时也加深读者对对象的层次结构的理解。

4.2.2　使用对象名称引用对象

每个对象都有一个名称，所以允许通过对象名称来引用对象。在 VBA 中引用工作簿时使用 WorkBooks("工作簿名称")这种格式，所以以下代码都表示工作簿引用。

Workbooks("Book1")——表示引用未保存过的工作簿“Book1”。

Workbooks("生产表.xlsm")——表示引用已保存过的工作簿“生产表.xlsm”，须有后缀名。

Workbooks("财务表.xls")——表示引用已保存过的工作簿“财务表.xls”，须有后缀名。

而使用以下方式引用工作簿必定出错。

Workbooks("C:\财务表.xls")——此方式引用将产生“下标越界”的错误提示。

从代码来看，代码编写者的目的应该是引用 C 盘中的文件，但 VBA 不允许此方式引用未打开的文件，必须在打开文件后再用“Workbooks("财务表.xls")”方式引用工作簿对象。而且即使是文件处于打开状态也不能采用此方式引用工作簿对象，VBA 会将“C:\”也当作工作簿名称的一部分，而事实上真实的文件名称中并未包含“C:\”。

注意：直接运行代码“Workbooks("生产表.xlsm")”或者“Workbooks("Book1")”一定会出错，不管当前是否已打开该名称的工作簿，因为它不是一句完整的代码。请查阅本书第 3.4.3 节。

如果需要引用插入到工作表中的图片，可以使用 Shapes ("图形名称")。例如以下 3 句代码可以分别引用 3 种图形对象。

Shapes("矩形 1")——引用工作表中的矩形对象，序号 1 表示它是第 1 个插入表中的。

Shapes("椭圆 2")——引用工作表中的椭圆对象，序号 2 表示它是第 2 个插入表中的。

Shapes("图片 3")——引用工作表中的图片，序号 3 表示它是第 3 个插入表中的。

当插入图形对象到 Excel 中时 Excel 会根据插入顺序对图形对象编号，图形的类型加编号就组成了该图形对象的完整名称。如果不知道某图形对象的类型和序号，那么可以选择该对象，然后查看名称栏中的图形名称，例如图 4-3 中选择了一个矩形，在名称栏中能看到它的全名。

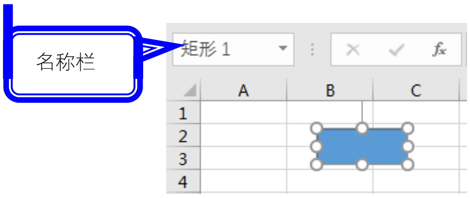


图 4–3　从名称栏查看图形对象的名称

如果需要引用指定名称的工作表，那么可以采用 Worksheets("工作表名称")这种格式。以下方式是常见的工作表引用。

Worksheets("Sheet1")——引用名为“Sheet1”的工作表。

Worksheets("财务工作表")——引用名为“财务工作表”的工作表。

Worksheets(2+4 & "月")——引用名为“6 月”的工作表，先运算加法表达式，然后将计算结果与月串连成一个字符串作为 Worksheets 的参数，从而引用该工作表。

4.2.3　使用复数形式表示对象集合

其实前面的引用中已经出现了对象的复数形式，不过使用了参数后只能引用参数所指定的单个对象。例如代码“Workbooks("生产表.xlsm")”表示引用单个工作簿“生产表.xlsm”，而 Workbooks 则表示当前打开的所有工作簿集合。

以下代码可以关闭所有打开的工作簿，不论打开的工作簿数量有多少。其中 Close 方法表示关闭工作簿。

```
Sub 关闭所有工作簿()　'放置位置：模块中
Workbooks.Close
End Sub
```

如果对复数形式的对象 Workbooks 使用参数，那么只引用对象集合中的单个对象。例如以下代码只能关闭名为“生产表.xlsm”的工作簿。

```
Sub 关闭生产表工作簿() '放置位置：模块中
Workbooks("生产表.xlsm").Close
End Sub
```

以下是复数形式引用对象集合的范例。

Shapes——表示图形对象集合。

Worksheets——表示工作表对象集合。

Names——表示名称对象集合。

Colors——表示颜色对象集合。

FormatConditions——表示条件格式对象集合。

PivotTables——表示数据透视表对象集合。

4.2.4　使用序号参数引用集合中的子对象

对于复数形式的对象集合，可以使用数字序号表示引用其中某一个子对象，示例如下。

Shapes(5)——表示引用图形对象集合中的第 5 个图形。

Worksheets(1)——表示引用第一个工作表，即最左边的一个。

Sheets(2) ——表示引用第二个表，表包含工作表，尽管图表也是表，但不属于工作表。

Names(Names.count)——表示最后一个名称，因为“Names.count”表示名称总数。

Cells(5) ——表示引用第 5 个单元格 E1（先行后列，所以 Cells(16386)表示 B2，因为 Excel 2010 中每行有 16384 个单元格，超过 16364 后从第二行开始计算）。

Cells(5,7) ——Cells 是比较特殊的单元格集合，既可使用单个参数，又可使用两个参数。当使用单个序号参数时，表示区域中按先行后列的位置序号获取的单元格对象，当使用两个参数时，第一参数表示行的位置，第二参数表示列的位置，所以 Cells(5,7)表示第 5 行第 7 列，即 G5。

需要特别指出的是当对象集合的参数是文本时，表示该参数是对象的名称；而参数是数值时，则表示参数是对象的序号。所以 Worksheets("2")和 Worksheets(2)的意义是完全不同的，前者表示引用名字是 2 的工作表，后者表示位置为 2 的工作表。

如果参数序号的值超过了对象集合的数量，那么将引用失败。

通过以下案例可以对集合与集合中的子对象有进一步认识。

假设图 4-4 中左图表示生产数据，若要求把产量大于等于 500 者标示为“达标”，那么可以采用以下代码。

```
Sub 标示是否达标() '放置位置：模块中
    Dim i As Byte     '声明一个 Byte 型的变量 i,作为循环语句中的变量
    For i = 1 To 9   '循环语句，从 1 到 9
   '如果第 i+2 行、第 2 列（产量从第 2 行开始，所以加 1）的值大于 500，那么将对应的复选框打勾
   '复选框当前的 Value 属性为 False,表示未打勾，设置为 True 表示打勾
      If Cells(i + 1, 2) >= 500 Then ActiveSheet. CheckBoxes (i).Value = True
    Next i
End Sub
```

代码运行效果如图 4-4 右图所示。



	A	B	C
1	姓名	产量	是否达标
2	赵文秀	541	☐达标
3	钱三明	507	☐达标
4	孙洪雷	516	☐达标
5	李大嘴	458	☐达标
6	周文明	460	☐达标
7	吴秀华	555	☐达标
8	郑英雄	402	☐达标
9	王维维	552	☐达标
10	冯年新	563	☐达标

	A	B	C
1	姓名	产量	是否达标
2	赵文秀	541	☑达标
3	钱三明	507	☑达标
4	孙洪雷	516	☑达标
5	李大嘴	458	☐达标
6	周文明	460	☐达标
7	吴秀华	555	☑达标
8	郑英雄	402	☐达标
9	王维维	552	☑达标
10	冯年新	563	☑达标

图 4-4　标示产量是否达标

Cells(i + 1, 2)表示单元格对象，它是 Cells 对象的子对象，表示第 *i*+1 行、第 2 列的单元格，即 *i* 的值是 1 时，那么 Cells(i + 1, 2)表示第 2 行第 2 列的单元格，即 B2；当 *i* 的值是 3 时，Cells(i + 1, 2)表示第 4 行第 2 列的单元格，即 B4。当 *i* 的值递增时，Cells(i + 1, 2)所引用的单元格对象也相应的由 B2 变成 B3、B4、B5……

代码中的 CheckBoxes 表示复选框集合，而 CheckBoxes(i)则表示集合中的子对象，当 i 为 1 时表示第 1 个复选框，i 为 2 时表示第 2 个复选框……

如果需要勾选工作表中所有复选框，那么删除“If Cells(i + 1, 2) >= 500 Then”部分即可，表示无条件逐个打勾。如果需要将工作表中所有复选框都去掉勾，那么去掉“If Cells(i + 1, 2) >= 500 Then”部分后再将 True 修改成 False 即可。

关于循环语句 For Next 的详细解释请参阅本书第 6 章。

本例案例文件请参考：..\ 第 4 章\ 4-1 引用集合中的子对象.xlsm

4.2.5 引用子对象

对象是有层级关系的，除了最底层的子对象，所有对象都有若干个子对象。例如一个工作簿中可能有 1 个、3 个或者 100 个工作表，一个工作表中可能有 1 个或者 50 个图形对象等，此处工作表是工作簿的子对象，图形对象是工作表的子对象。

对象与下层对象的表示方式为“对象.子对象”。

例如引用“生产表.xlsm”中的“1 月”工作表，代码如下:

```
Workbooks("生产表.xlsm").Worksheets ("1 月")
```

如果引用“财务报表”工作表中的第二个图表，则采用以下代码：

```
Worksheets(“财务表").ChartObjects("图表  2")
```

在引用多级对象时要注意它们的层级关系，例如图形对象是工作表的子对象，工作表是工作簿的子对象，所以不能将图形对象作为工作簿的子对象来调用。所以通过以下代码计算工作簿中图形对象的数量必将执行失败。

```
Sub  图形对象数量() '放置位置：模块中
MsgBox Workbooks("工作簿 1.xls").Shapes.Count
End Sub
```

如果改用以下代码计算指定工作簿中第 1 个工作表的图形对象数量就不再有问题：

```
Sub  图形对象数量() '放置位置：模块中
MsgBox Workbooks("工作簿 1.xls").Worksheets(1).Shapes.Count
End Sub
```

4.2.6 引用活动对象

VBA 专门给活动对象赋予了一个特别的称谓，通常使用“Active”表示活动对象。例如活动工作表用“ActiveSheet”表示，活动工作簿用“ActiveWorkbook”表示，活动窗口用“ActiveWindow”表示，活动单元格用“Activecell”表示……

值得注意的是，任何对象的活动成员都只有一个，所以在名称中 Activesheet、ActiveWorkbook 和 Activecell 等都不使用复数形式，也没有参数。

根据以上规则，读者可以举一反三，活动图表用“ActiveChart”表示，活动打印机用“ActivePrinter”表示，但没有活动图形或者活动透视表的说法。

当调用活动对象的子对象时，可以忽略活动对象名称，直接引用其子对象。

例如调用活动工作簿中的“总表”工作表，可以采用以下两种方式：

```
ActiveWorkbook.Worksheets ("总表")
Worksheets ("总表")
```

事实上第二种写法不仅写法上更简单，它的执行效率也高于第一种，因为第一种方法其实需要调用两个对象，而第二种方法只调用一个对象，速度稍快些。

同时，调用活动工作表中的 B 列也有以下两种写法：

```
ActiveSheet.Range("B:B")
```

```
Range("B:B")
```

注意：本节所提到的 Workbooks、Worksheets、Sheets、Range、ChartObjects 等对象的功能和语法在后面的章节会有专门的说明，读者也可以查阅帮助了解这些信息，帮助中有每个对象的说明，以及该对象相关的方法、属性等相关知识。

4.2.7　引用父对象

与子对象相对的是父对象，即某个对象的上一层对象。

表示子对象时，采用小圆点并在后面连接子对象名称的方式；而表示父对象时，则用一个称谓——Parent。因为一个对象的子对象有可能是一个也可能有多个，但是对象的父对象只能有一个，所以统一用 Parent 表示。以下是父对象的应用。

Range("A1").Parent.Name——用于获取单元格所在工作表的名称，因为单元格的父对象是工作表。

ActiveSheet.Shapes("矩形 1").Parent.Parent.Name——用于获取“矩形 1”图形对象所在工作簿的名称，因为图形对象的父对象是工作表，工作表的父对象是工作簿。

Range("A5").Comment.Parent——用于获取单元格的批注的父对象，也就是单元格本身。

以下案例“获取有批注的单元格地址”是上述内容的应用，代码中除父对象的应用外，还涉及 Comment、Erro 和防错语句等，这些知识点将在后面的相关章节有详细阐述，此处仅需掌握父对象的调用方式即可。

```
Sub 获取有批注的单元格地址() '放置位置：模块中
'声明两个变量，前者代表批注，后者代表每个批注所在单元格有地址
Dim    Com As Comment, ComAddress As String
For Each Com In ActiveSheet.Comments    '遍历当前工作表中所有批注（循环语句的应用）
'将批注所在单元格的地址与变量串连起来，形成一个字符串，中间用换行符隔开
        ComAddress = ComAddress & Chr(13) & Com.Parent.Address
    Next Com   '下一个批注
    '如果变量 ComAddress 的长度大于 0（表示找到了批注），那么提示批注所在单元格的地址
    If Len(ComAddress) > 0 Then MsgBox ComAddress Else MsgBox "没找到批注"
End Sub
```

执行以上代码后可以将活动工作表中所有批注所在单元格的地址一并罗列出来。例如在图 4-5 中，工作表中有 3 个批注，因此执行结果是在信息框中罗列出 3 个批注所在单元格的地址。

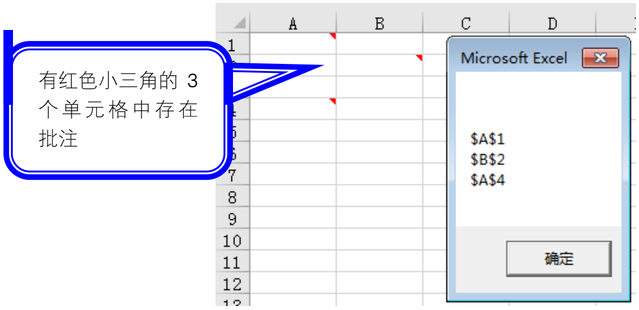


图 4-5　获取所有批注的地址

在本例的代码中，变量 Com 代表批注，Com.Parent.Address 代表批注的父对象的单元格地址，而变量 ComAddress 用于存放所有找到的批注所在单元格的地址，将使用 chr(13)将它们串连起来。其中 Chr(13)是换行符，在本例中表示每一个批注所在单元格的地址占据一行。

循环语句的语法将在本书第 6 章详细介绍，此处了解父对象的用法即可。

本例案例文件请参考：..\ 第 4 章\ 4-2 父对象的应用.xlsm

4.2.8 利用 With 语句引用重复出现的对象

当一段代码中重复出现某个对象时，利用 With 语句只书写一次对象名称即可，从而简化代码，同时提升程序执行效率。

例如在设置单元格的多种格式时，普通用法是每设置一种格式需要引用一次对象，代码如下：

```
Sub 设置格式()                                     '放置位置：模块中
    Range("A1").Font.Name = "方正姚体"             '设置 A1 单元格的字体名称为方正姚体
    Range("A1").Font.Size = 10                     '设置 A1 单元格的字体大小为 10 号
    Range("A1")..Font.Bold = True                  '将 A1 单元格字体设置为加粗效果
    Range("A1").Font.Color = 255                   '设置 A1 单元格的字体颜色为 255，即红色
End Sub
```

此编写方式很直观，能清晰地表达出对象属性的赋值过程，便于阅读，然而在编写时会比较烦琐，需要反复地书写同一个对象。其次在执行效率上也较差，需要多次引用对象，从而消耗更多内存资源和时间。

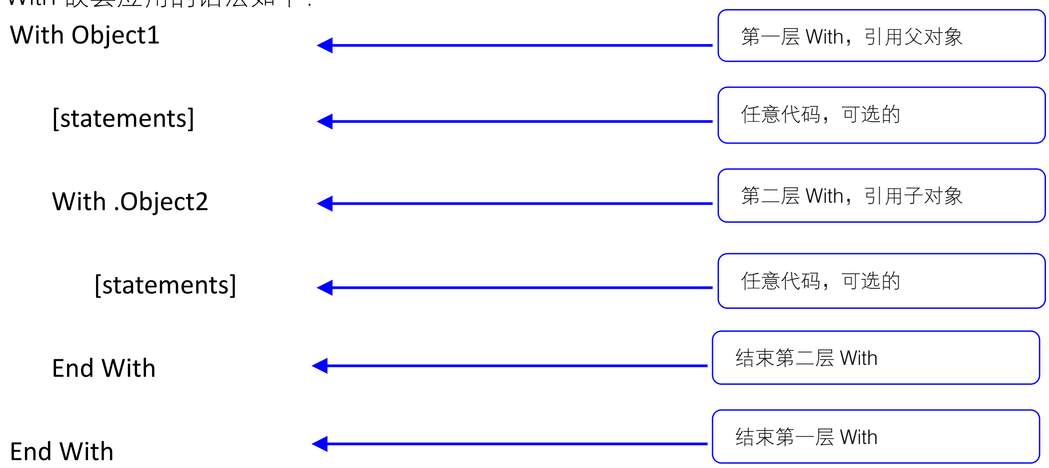
如果采用 With 语句，代码将得到极大简化：

```
Sub 设置格式 2()                                   '放置位置：模块中
    With Range("A1").Font                          '引用 A1 单元格的字体对象
        .Name = "方正姚体"                         '将字体名称设置为方正姚体
        .Size = 10                                 '字体的大小设置为 10 号
        .Bold = True                               '将字体设置为加粗效果
        .Color = 255                               '将字体颜色设置为 255，即红色
    End With                                       '结束 With 语句
End Sub
```

在利用 With 语句引用对象时，应将代码缩进并对齐，从而让代码具有层次感，有利于阅读和理解代码。

对于以上代码，如果添加两个要求：在 A1 单元格录入文本 “VBA 效率之源”，同时将单元格文字居中显示，那么该如何处理 With 语句呢？此时需要用到 With 语句的嵌套知识。

With 嵌套应用的语法如下：



其中“Object1”表示父对象，“.Object2”表示子对象，即“Object2”是“Object1”的子对象。如果忽略“.Object2”中的小圆点，那么两个对象之间将不存在从属关系。

言归正传，回到前面的需求，利用 With 嵌套设置格式及赋值的代码如下：

```
Sub 设置格式 3()                                  '放置位置：模块中
    With Range("A1")                              '引用单元格对象 A1
        .Value = "VBA 效率之源"                    '对单元格赋值
        .HorizontalAlignment = xlCenter           '让单元格居中对齐，“xlCenter”代表居中
        With .Font                                '引用 A1 单元格的字体对象
            .Name = "方正姚体"                     '设置字体名称为方正姚体
            .Size = 10                            '设置字体的大小为 10 号
            .Bold = True                          '将字体设置为加粗效果
            .Color = 255                          '设置字体颜色编号为 255，即红色
        End With                                  '结束里层 With 语句
    End With                                      '结束外层 With 语句
End Sub
```

在以上代码中，外层 With 引用的对象是 Range("A1")，其后两句代码的赋值对象就是 Range("A1")。里层 With 引用的对象是“.Font”，此处的小圆点表示 Font 是 Range("A1")的子对象。在“With.Font”语句之后的所有带有前置小圆点的代码的操作对象其实都是“Range("A1") .Font”，例如“.Name”的作用相当于“Range("A1") .Font. Name”。

对于有 With 嵌套的代码需要注意代码的缩进与对齐，通过代码的层次感能更方便地查阅代码，了解对象与对象、对象与属性间的关系。

以下是两种错误的 With 用法：

```
Sub 设置格式 4()                                  '放置位置：模块中
    With Range("A1")                              '引用单元格对象 A1
        .Value = "VBA 效率之源"                    '对单元格赋值
        .HorizontalAlignment = xlCenter           '让单元格居中对齐
        End With                                  '结束 With 语句用
        With .Font                                '引用 A1 单元格的字体对象
            .Name = "方正姚体"                     '设置字体名称
            .Size = 10                            '设置字体的大小
            .Bold = True                          '将字体设置为加粗效果
            .Color = 255                          '设置字体颜色
    End With                                      '结束 With 语句
End Sub
```

在上述代码中，第一个 End With 结束了 Range("A1")对象的引用，所以后面的代码“With .Font”在引用子对象时必定出错，它必须位于外层的 With 与 End With 语句之间。

```
Sub 设置格式 5()                                  '放置位置：模块中
    With Range("A1")                              '引用单元格对象 A1
        .Value = "VBA 效率之源"                    '对单元格赋值
        .HorizontalAlignment = xlCenter           '让单元格居中对齐
        With Font                                 '引用字体对象
            .Name = "方正姚体"                     '设置字体名称
            .Size = 10                            '设置字体的大小
            .Bold = True                          '将字体设置为加粗效果
            .Color = 255                          '设置字体颜色
```

```
            End With                        '结束 With 语句
        End With                            '结束 With 语句
    End Sub
```

上述代码中子对象 Font 未添加前置小圆点，所以在对 Font 对象的 Name 属性赋值时将出错并提示“要求对象”。

本例案例文件请参考：..\ 第 4 章\ 4-3 With 语句.xlsm

对于以上六种对象的表示方式，看起来有些枯燥乏味，实际上掌握这些知识对于编程有着深远的影响。因为 VBA 的代码就是用来操作对象的，而对象是 VBA 中最重要的一个知识点，由对象牵引出对象的属性、对象的方法、对象的事件等，这些知识占 VBA 整体知识数量的 80%以上，所以有必要了解对象的表示方法。

4.3　Range 对象

Range 对象也被称为单元格对象，是 Excel 中应用最频繁的对象。本节展示 Range 对象的各种引用方法，为后续编写复杂程序打下坚实的基础。

4.3.1　Range("A1")引用方式

Range ("A1")样式引用单元格或者区域，实质上是利用 Range 将文本地址转换成对象引用，类似于工作表函数 Indirect。

1．引用单元格

在用 Range 引用单元格时，代表单元格地址的文本需采用列标加行号的形式来呈现地址，而且需要使用半角的双引号。以下是一些常见的单元格引用形式。

Range ("A1")——表示将字符串“A1”转换成 A1 单元格的引用，A 为列标，1 为行号。

Range ("Z500")——表示将文本“Z500”转换成单元格 Z500 的引用。

Range ("VF56980")——表示将文本“VF56980”转换成单元格引用，对 Excel 2003 无效,因为 Excel 不存在 VF 列，最大只有 IV 列。

以下形式的引用是错误的。

Range (“A1”)——参数不允许使用全角的引号。

Range("ZZZ500")——列标超出允许的范围，Excel 2003 的列标上限为 IV 列，Excel 2016 的列标上限为 XFD 列。

Range("B&I+5")——表达式不能放在引号中间，改为 Range("B" & I+5)即可。

Range(B5) ——参数需要添加引号。

单元格对象有很多属性，Value 属性是它的默认属性，表示单元格的值。VBA 允许在调用默认的属性时忽略属性名称，所以引用 Range ("A1").Value 时可以书写为 Range ("A1")。例如：

Msgbox Range ("A1")——虽然没有指定属性，但是在对话框中仍然显示 A1 单元格的值；

MsgBox Range("A1").Address——表示获取单元格的地址属性，必须指明属性名称。

注意：当 Range 的参数中有字母时，字母不区分大小写。事实上在 VBA 中绝大部分情况下参数名称都不区分字母大小写。不过在使用比较运算符比较字母大小时会区分大小写。

2．引用行与列

Range 引用整行或者整列与工作表函数引用整行整列时采用的规则一致——行号加冒号加行号，或者列标加冒号加列标，不过 Range 的参数是文本，需要添加半角双引号。

以下是整行、整列引用的示例。

Range("B:B")——表示引用 B 列，其中引号必须在半角状态下录入，冒号则不受限制。

Range("B:Z")——表示引用 B 到 Z 列，总共 25 列。

Range("3:20")——表示引用第 3 到 20 行，总共 18 行。

事实上对于整行、整列的引用，采用 Rows 和 Columns 更容易理解，Rows 用于引用行，Columns 用于引用列，示例如下。

Rows("2:2")——表示引用第 2 行，必须使用引号。

Rows(2) ——表示引用第 2 行，可以不使用引号。

Rows("7:" & 7+5) ——表示引用第 7 行和其后面的 5 行，一共 6 行，即引用 7 到 12 行。

Columns("C:C")——表示引用第 C 列，采用列标作参数；

Columns(3)——也表示引用第 C 列，采用数字作参数，表示顺序，即从左到右第 3 列。

Rows 和 Columns 分别表示行对象集合与列对象集合，包括 A1~XFD1048576 共 17179869184 个单元格。Range("1:1048576")也包括 A1:XFD1048576 共 17179869184 个单元格，但是它们三者还是有区别的，主要体现在子对象上面。换言之，Rows、Columns 和 Range(“1:1048576”)三者代表相同对象，但是三者的子对象是不同的。

Range("1:1048576")(1) ——表示 A1 单元格。

Columns(1) ——表示第 1 列。

Rows(1) ——表示第 1 行。

注意：测试引用是否正确时常用代码 MsgBox Range("A1").Value 来验证，但是 MsgBox 一次只能显示一个值，而引用行、列时包含多个单元格，因此不能使用以下代码。

```
MsgBox Range("A:A").Value
MsgBox Columns(3).Value
```

验证引用方式是否正确应该是检查它的地址，代码如下：

MsgBox Range("2:4").Address——获取列对象的地址，Range 对象不管包含多少单元格，它的地址总是单个值，可以通过 Msgbox 函数显示出来。

3．引用区域

区域是指多个连续单元格组成的矩形区域，只要指定矩形的两个对角单元格就可以确定区域大小。例如矩形区域的左上角单元格加右下角单元格或者左下角的单元格加右上角单元格。

使用 Range 引用一个区域包含两种方式，一是采用单元地址与冒号形成区域，具体样式为：

```
Range("左上角单元格地址:右下角单元格地址")
Range("左下角单元格地址:右上角单元格地址")
```

以下是常见的引用案例。

Range("A1:C5")——表示引用 A1 到 C5 的矩形区域，包括 15 个单元格。

Range("F3:G1")——表示引用 F1 到 G3 的矩形区域，VBA 会自动将不规则的地址转换成规则的编码方式。

Range("C1:C1048576")——表示引用 C 列第 1 到第 1048576 行，也就是整个 C 列。

Excel 2003 中行数上限是 65536，所以上面的代码只能用于 Excel 2007 及更高版本。

区域总是包含多个单元格，所以可以通过索引号从区域中引用单个单元格。

Range("A1:C5")(1)——表示 A1:C5 区域中左上角的单元格 A1。

Range("F5:G8")(7) ——表示 F5:G8 区域中第 7 个单元格，按从左到右、从上到下的顺序计算序号。

Range("F2:G3")(5) ——表示 F2:G3 区域中第 5 个单元格 F4。由于 F2:G3 区域中只有 4 个单元格，所以序号 5 则表示从 Range("F2:G3")区域下方一行 F3:G3 中继续取值。图 4-6 展示了 Range("F2:G3")(5)的序号取值过程。

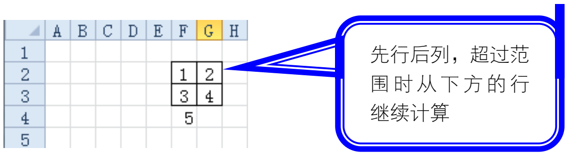


图 4–6 Range("F2:G3")(5)示意图

Range 引用区域的第二种方式是采用两个单元格或者区域作参数形成的区域，新的区域是包含两个参数的最小矩形区域，它以参数中所有单元格对象的最小行号和最大行号作为上下边界，以参数中所有单元格的最小列号和最大列号作为左右边界。

具体样式为：Range(Cell1, Cell2)

以下代码表示由 D3 和 I5 两个单元格形成的矩形区域，效果如图 4-7 所示。

```
Range(Range("D3"), Range("I5"))
```

以下代码表示由 C3:C4 和 H2:J2 两个区域形成的新的区域，效果如图 4-8 所示。

```
Range(Range("C3:C4"), Range("H2:J2"))
```

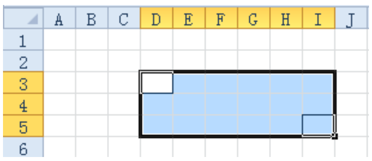


图 4–7 Range(Range("D3"), Range("I5"))

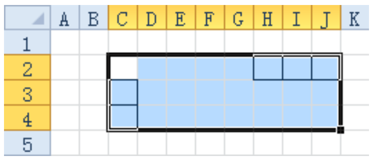


图 4–8 Range(Range("C3:C4"), Range("H2:J2"))

4．引用多区域

Range 可以同时引用多个区域，其语法为：

```
Range ("区域 1,区域 2,区域 3…")
```

在使用 Range 引用多区域时，区域与区域地址之间采用逗号分开，然后在前后分别使用半角的双引号。它不限制区域个数，但是 Range 的参数却有长度限制——不超过 256 个字符。

Range("C4:C11,E4:E11")——表示同时引用 C4:C11 和 E4:E11 两个区域。

Range("D5,E4:E11,F10")——表示同时引用 D5、E4:E11 和 F10 三个区域。

Range("6:6,C:C")——表示同时引用第 6 行和第 C 列，此形式引用允许多区域重叠。

4.3.2 Cells(1,1)引用方式

Cells 表示工作表中所有单元格的集合，而使用行坐标和列坐标可以访问集合中任意一个子对象，其中行坐标必须是数值，列坐标可用数值也可用列标字母。

当指定工作表名称时，Cells 能引用指定工作表的单元格，当忽略工作表名称时表示引用活动工作表的单元格，示例如下。

Cells(1,1) ——表示引用工作表中第 1 行、第 1 列的单元格，即 A1。

Cells(100,10*5) ——表示引用第 100 行、10×5 列的单元格，两个参数都可以使用表达式。

Cells(5, "R")——表示引用第 5 行第 R 列，即 R5 单元格;

WorkSheets("生产表").Cells(500, "BC")——表示引用生产表中的 BC500 单元格。

Cells(1,1)形式只能引用单个单元格，引用区域时应改用 Range。

Cells 还有一种较特殊的用法——作为另一个 Range 对象的子对象。当 Cells 的父对象是区域时，Cells 代表区域中的所有单元格，而非工作表中的所有单元格。

以下代码表示引用 B3:H5 区域第 2 行第 2 列的值，图 4-9 能较好地说明 Cells 的工作原理。

```
Range ("B3:H5").Cells(2,2)
```

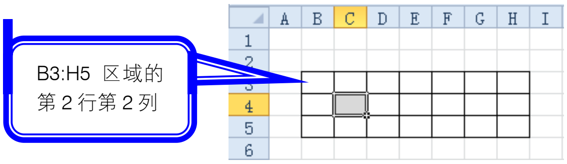


图 4-9　Range ("B3:H5").Cells(2,2)

乍一看，估计所有读者都会认为“Range ("B3:H5").Cells(2,2)”这种形式引用单元格没有实用性，但在实际工作中经常会用到。例如对 B3:H5 区域先列后行的方式逐个单元格检查拼写，可用以下代码。

```
Sub 检查拼写()  '放置位置：模块中
'声明两个 Integer 型的变量，用于纵向循环和横向循环
    Dim RowNumber As Integer, ColNumber As Integer
    '从第 1 行到 Range("B3:H5")区域中的最后 1 行，其中 Rows.Count 表示总行数
    For RowNumber = 1 To Range("B3:H5").Rows.Count
        '从第 1 列到 Range("B3:H5")区域中的最后 1 列，其中 Columns.Count 表示总列数
        For ColNumber = 1 To Range("B3:H5").Columns.Count
            '逐个检查区域中的单元格拼写是否正确。
'Cells(RowNumber, ColNumber)表示引用区域中指定坐标的单元格
            Range("B3:H5").Cells(RowNumber, ColNumber).CheckSpelling
        '执行下一次循环（此处表示判断下一行）
        Next   RowNumber
    '执行下一次循环（此处表示判断下一列）
    Next   ColNumber
End Sub
```

此代码分别检查 B3、B4、B5、C3、C4、C5…H5 单元格的拼写是否正确。当 Cells 的参数使用变量时，用 Cells 作为 Range 对象的子对象进行循环相当有用，可以随意访问区域中任意的子对象。

本例案例文件请参考：..\ 第 4 章\ 4-4 Range.cells 应用.xlsm

本例中的变量的应用请参考本书 5.2 节，For Next 循环语句的具体用法与含义请参考本书 6.2 节。

4.3.3　[A1]引用方式

[A1]引用方式和 Range("A1")引用方式有异曲同工之妙，都能将文本形式的单元格地址转换成引用，不过它们的差异也不少。

相同点表现在两者都能将单元格地址转换成引用，且支持多区域，示例如下。

[A1:A100]、[B5]、[F:F]、[F:F,G10]、[A:H,2:4]——[A1]形式的引用。

Range("A1:A100")、Range("B5")、Range("F:F")、Range("F:F,G10")、Range("A:H,2:4")——Range形式的引用。

以上两种表达方式能产生相同引用，但是两者的差异也很明显。

首先，Range 的参数需要采用引号，而[A1]则不需要引号，直接将文本转换成引用。

例如以下代码可返回 True，表示两者的引用对象的地址一致，其中 Address 表示单元格的地址。

```
MsgBox [A1:B5].Address = Range("A1:B5").Address
```

如果[A1]形式的引用采用了引号，那么它不再是对象，而是字符串，示例如下。

["A1"]——表示字符串 A1，它不再具有 Range 对象的属性；

[A1] ——表示单元格 A1，是一个对象，具有单元格的一切属性。

其次，录入 Range("A1")形式的对象时有快速提示信息，而[A1]形式则没有，所以在编写代码时尽量采用 Range("A1")形式。图 4-10 是 Range("A1")的快速信息列表。

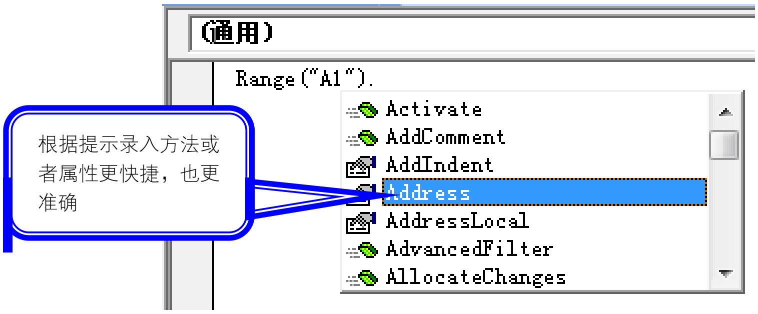


图 4–10　快速信息

最后，[A1]形式的引用不支持参数，示例如下。

Range("A1:B10")(5)——表示引用 A3 单元格。

[A1:B10](5)——表示无法引用成功，将出现“错误的参数号”提示。

符号“[]”还有着转换运算表达式的功能，例如：

```
MsgBox [sum(a1,20,b15*2)]
```

此句代码可以得到公式 sum(a1,20,b15*2)的运算结果，相当于在单元格中录入公式“=sum(a1,20,b15*2)”。

```
MsgBox [=sum(row(1:100))]
```

此句代码用于计算 1 到 100 之和，“[]”符号中间的表达式完全遵循公式的规则，而不采用VBA 的语法。

4.3.4　活动单元格：ActiveCell

在 VBA 中采用 ActiveCell 表示活动单元格。活动单元格 ActiveCell 和选区 Selection 有不同的含义。Selection 表示当前选择的区域，可能是单个单元格也可以是多个单元格，当选择的区域只有单个单元格时，选区 Selection 才等于活动单元格 ActiveCell。

以下代码足以说明 Selection 与 ActiveCell 的差异：

```
Range("A1:B5").Select
ActiveCell.Value = "VBA 也疯狂"
```

首句代码可以选择 A1:B5 区域，当执行此语句后，Selection 即代表 A1:B5 区域，而此时的活动单元格 ActiveCell 是选区中左上角的 A1。所以执行两句代码的结果是选择 A1:B5 区域，并在 A1 单元格赋值，如图 4-11 所示。

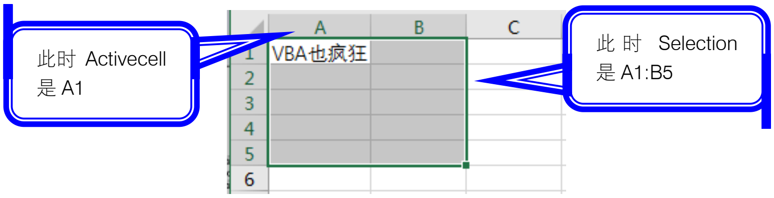


图 4–11　ActiveCell 与 Selection 的区别

如果要改变活动单元格，可使用 Range.Select 方法或者 Range.Activate 方法。前者表示选择单元格，当选区发生变化时，活动单元格相应变化，不过后者更为可靠。

例如要将 F5 单元格设置为活动单元格，那么可用以下代码。

```
Range("F5").Activate
```

Range.Select 和 Range.Activate 都不能跨表，仅作用于活动工作表。

当活动工作表是第 1 个表时，以下代码将执行失败。

```
Worksheets(2).Range("A5").Activate   '激活第 2 个工作表的 A5 单元格
```

改用两句才能达成需求，先选择第 2 个工作表，然后再激活 A5 单元格。

```
Worksheets(2).Select    '选择第 2 个工作表
Range("a5").Activate    '激活 A5 单元格
```

4.3.5　屏幕坐标下的单元格：RangeFromPoint

RangeFromPoint 是窗口对象 Window 的一个方法，完整书写方式是 Window.RangeFromPoint，它可以获取位于屏幕上指定坐标位置的 Shape 或 Range 对象。如果指定坐标位置上没有任何形状，那么此方法将返回 Nothing；如果在指定坐标的单元格上方有 Shape 对象，那么此方法返回一个 Shape 对象；如果指定坐标处只有单元格，那么返回 Range 对象。

在图 4-12 中，鼠标指针下是图片，所以此时该坐标下的 Window.RangeFromPoint 对象就是 Shape 对象，即工作表中的图片。如果删除图片，那么此时该坐标下的 Window.RangeFromPoint 对象就是 Range 对象，即 B2 单元格。

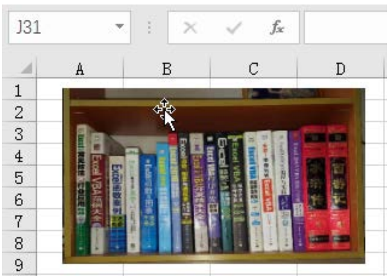


图 4–12　鼠标指针下的对象

RangeFromPoint 方法的具体语法如下：

```
Window.RangeFromPoint(X,Y)
```

其中参数 X 表示屏幕的纵坐标，Y 表示屏幕的横坐标，单位皆为像素，坐标原点是屏幕左上角。例如需要获取横坐标 250、纵坐标 250 处的对象名称，可使用以下过程。

```
'代码思路分析
'首先利用 Window.RangeFromPoint 方法获取横坐标与纵坐标皆为 250 处的对象，将它赋值给变量 Obj
'然后利用条件语句 If 判断是否存在对象，即坐标下方是否有单元格或者图形对象。Is Nothing 表未什么也没有
'如果存在对象的话，那么再通过 TypeName 函数判断对象的类型，如果类型是 Range 则提取它的单元格地址
'否则提取它的名称（表示这是一个图形对象）
Sub 获取坐标下的对象名称()'放置位置：模块中
  Dim Obj As Object        '声明一个 Object 型的对象变量
  Set Obj = ActiveWindow.RangeFromPoint(250, 250) '将 RangeFromPoint 获得的对象赋予变量
  If Obj Is Nothing Then   '如果对象不存在
    MsgBox "没有单元格和图形对象", vbOKOnly, "友情提示" '提示没有对象
  Else                     '否则（表示该坐标下方有单元格对象或者图形对象）
    '如果对象的类型是 Range，那么提示对象的地址，否则提示对象的名称（图形对象）
    If TypeName(Obj) = "Range" Then MsgBox Obj.Address Else MsgBox Obj.Name
  End If
End Sub
```

由于案例中坐标 250*250 处是图片，图片的名字是“我的图书”，因此执行以上过程后会在信息框中生成“我的图书”；如果删除图片再执行过程，那么将在信息框中生成单元格 B2 的地址。

为了提升代码的通用性，此过程较长。首先需要判断在指定坐标处是否有对象，还需要判断对象是 Range 对象还是 Shape 对象。如果确定该坐标处只有 Range 对象，那么代码就可以大大简化，一句代码即可解决：

```
MsgBox ActiveWindow.RangeFromPoint(250, 250).Address
```

在本例的过程中，RangeFromPoint 获得的是一个对象，但不确定是 Range 对象还是 Shape 对象，因此利用 TypeName 函数作判断。

关于变量、变量类型和 TypeName 函数的具体用法，请参阅本书第 5 章，条件语句 If Then 的用法则在第 6 章详细阐述。

本例案例文件请参考：..\ 第 4 章\ 4-5 获取坐标下的对象名称.xlsm

RangeFromPoint 在实际工作中用处不大，而且即使要用也需要配合 API 函数使用，通常是通过 API 函数获取鼠标指针移动时的坐标，RangeFromPoint 方法通过该坐标获取鼠标指针下方的单元格或者图片。本书不涉及 API 知识，因此不演示 RangeFromPoint 与 API 的应用，不过可以为读者提供一个案例，供大家在休闲娱乐的同时了解 API 与 RangeFromPoint 搭配应用时产生的奇效。本书案例文件中“..\ 第 4 章\鼠标移过单元格时在状态栏显示地址.xlsm”的文件中采用了 API 配合 RangeFromPoint 应用，实现鼠标指针移过单元格时在状态栏显示单元格的地址。

4.3.6 选区：Selection、RangeSelection

Selection 表示当前选中的对象，可能是单元格也可能是图形对象、图表。即当前选中的对象是什么 Selection 就代表什么。

RangeSelection 表示指定窗口中工作表上的选定单元格。换言之 Selection 代表当前选定的对象，而 RangeSelection 只代表选定对象中的单元格，如果当前选中的活动对象是图片或者图表，那么 RangeSelection 代表上一次选择的 Range 对象。

可以通过以下步骤区分 Selection、RangeSelection。

（1）新建一个空白工作表，然后插入一张图片。

（2）选择 A 列，再选择图片，如图 4-13 所示。

（3）按【Alt+F11】组合键打开 VBE 窗口，选择菜单“插入”→“模块”。

（4）在模块中录入以下代码。

```
Sub 区分 Selection 和 RangeSelection() '放置位置：模块中
 '提示 Selection 的名称和 RangeSelection 的地址
 MsgBox "Selection：" & Selection.Name & Chr(13) & _
 "RangeSelection：" & ActiveWindow.RangeSelection.Address, vbOKOnly, "友情提示"
End Sub
```

此代码用于提示 Selection 的名称和 RangeSelection 的地址。

（5）选择代码，按下【F5】键执行过程，VBA 将弹出如图 4-14 所示的提示框，从图中可以看到当前的选择对象名称是“Picture 1”，而选中的单元格对象地址是“$A:$A”，表示当前状态下 Selection 和 RangeSelection 两者并不一致。

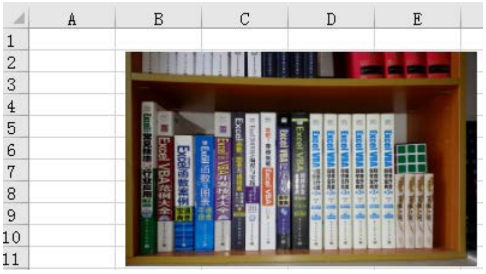


图 4–13　选择插入的图片再执行代码



图 4–14　Selection 与 RangeSelection 的分别

如果当前选择的对象是单元格或区域，那么 Selection 和 RangeSelection 将表示相同对象。实际工作中，如果要引用当前选择的区域，则尽量使用 RangeSelection 而不是 Selection，只有确定选择对象是单元格以外的对象时才用 Selection。

如果需要确定当前活动的对象是否为单元格，可以使用 TypeName 函数来判断。

```
Sub 判断当前活动对象是否为单元格() '放置位置：模块中
   MsgBox TypeName(Selection) = "Range"
End Sub
```

TypeName 函数的功能是判断其参数的类型，计算结果要区分大小写。所以上述代码中“Range”的首字母必须大写。在 TypeName(Selection)计算的结果为“Range”时，表示当前活动的对象是单元格。

本例案例文件请参考：..\ 第 4 章\ 4-6 Selection 与 RangeSelection.xlsm

4.3.7　已用区域：UsedRange

UsedRange 是 WorkSheet 对象的一个属性，也是 WorkSheet 的子对象，它表示工作表的已用数据区域。已用数据区域是指工作表中包括所有已经使用过的单元格的最小矩形区域，而已经使用的区域包括有数据的单元格和有格式信息的单元格。某些单元格在删除数据后保留了单元格格式信息，所以虽然看起来是空白单元格，但也属于已用区域。

以图 4-15 的数据为例，此工作表的已用区域是 A1:F9，因为包含所有使用过的单元格的最小区域是 A1:F9。

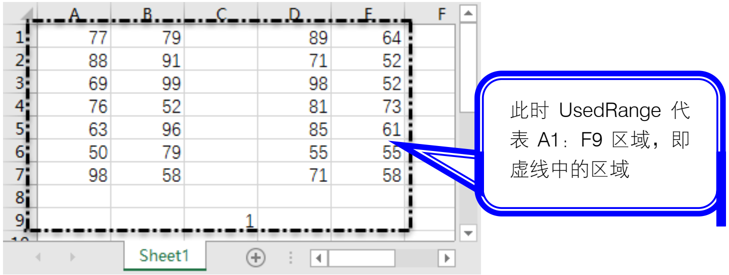


图 4–15　随机录入数据用于测试已用区域范围

UsedRange 是一个很重要的对象，在编程中经常需要使用。

在图 4-16 中，由于列宽未设置好导致部分列的数据未完整地显示出来，而部分列的宽度又太大，显得不够美观。此外还要求对工作表的数据区域添加边框，以及设置格式为居中对齐。

实现以上需求可以采用以下代码：

```
Sub 对已用区域添加格式()                        '放置位置：模块中
    With ActiveSheet.UsedRange                  '引用活动工作表的已用数据区域
        .HorizontalAlignment = xlCenter         '设置格式为居中对齐显示
        .EntireColumn.AutoFit                   '自动调整列宽
        .Borders.LineStyle = xlContinuous       '添加边框
    End With
End Sub
```

代码执行效果如图 4-17 所示。

	A	B	C	D
1	姓名	产量	不良品	不良率
2	赵洪伟	225	9	4.0%
3	陈金来	232	20	8.6%
4	吴鑫	239	11	4.6%
5	赵秀文	285	10	3.5%
6	朱明	211	20	9.5%
7	陈冲	258	5	1.9%
8	朱贵	231	8	3.5%
9	陈丽丽	209	8	3.8%
10	欧阳正雄	216	6	2.8%
11	刘子中	207	11	5.3%

图 4–16　未设置格式的产量表

	A	B	C	D
1	姓名	产量	不良品	不良率
2	赵洪伟	225	9	4.0%
3	陈金来	232	20	8.6%
4	吴鑫	239	11	4.6%
5	赵秀文	285	10	3.5%
6	朱明	211	20	9.5%
7	陈冲	258	5	1.9%
8	朱贵	231	8	3.5%
9	陈丽丽	209	8	3.8%
10	欧阳正雄	216	6	2.8%
11	刘子中	207	11	5.3%

图 4–17　设置格式后的效果

过程中需要引用三次“ActiveSheet.UsedRange”，所以可以使用 With 语句简化引用。

HorizontalAlignment 表示居中对齐方式，将其赋值为 xlCenter 表示居中显示。

EntireColumn 表示整列，通过 Range.AutoFit 方法自动调整列宽时仅对整列或者整行生效，而本例中的 UsedRange 仅包含 11 行，所以需要使用 EntireColumn 将它重置为整列后再调整列宽。如果要求是将已用数据区域自动调整行高，那么可以采用以下语句：

```
ActiveSheet.UsedRange.EntireRow.AutoFit        '自动调整行高
```

Range.Borders 表示单元格的所有边框。LineStyle 表示线型，xlContinuous 表示实线，所以“.Borders.LineStyle = xlContinuous”这句代码表示对选择区域添加实线边框。

在本过程中，UsedRange 扮演着极其重要的角色，可以自动计算需要设置格式的区域大小，避免手动查找起始行、起始列、结束行与结束列。

注意：任何时候 UsedRange 的父对象都不能省略，正如引用图形对象 Shape 时不能忽略其父对象一样。在本例中 UsedRange 的父对象是活动工作表。

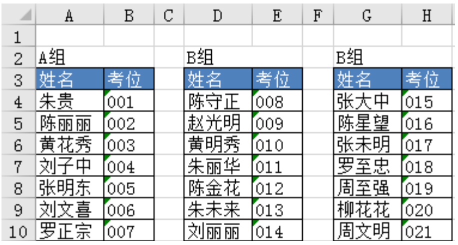
本例案例文件请参考：..\ 第 4 章\ 4-7 对已用区域设置格式.xlsm

4.3.8　当前区域：CurrentRegion

CurrentRegion 是 Range 对象的一个属性，也是 Range 的子对象，它表示单元格的当前区域。当前区域是指包含当前单元格且以空行与空列的组合为边界的区域，例如在图 4-18 中，B3 单元格的当前区域是 A2:B10，D3 单元格的当前区域是 C2:D10，而 H3 单元格的当前区域是 G2:H10。可以使用以下代码验证。

```
Sub 获取单元格的当前区域地址() '放置位置：模块中
    '分别计算 3 个单元格的当前区域地址，用回车符分隔开并显示在信息框中
    MsgBox Range("B3").CurrentRegion.Address & Chr(13) _
    & Range("D3").CurrentRegion.Address & Chr(13) _
    & Range("H3").CurrentRegion.Address
End Sub
```

执行以上代码后将返回如图 4-19 所示的结果。



	A	B	C	D	E	F	G	H
1								
2	A组			B组			B组	
3	姓名	考位		姓名	考位		姓名	考位
4	朱贵	001		陈守正	008		张大中	015
5	陈丽丽	002		赵光明	009		陈星望	016
6	黄花秀	003		黄明秀	010		张未明	017
7	刘子中	004		朱丽华	011		罗至忠	018
8	张明东	005		陈金花	012		周至强	019
9	刘文喜	006		朱未来	013		柳花花	020
10	罗正宗	007		刘丽丽	014		周文明	021

图 4–18　考位排列表

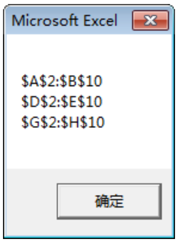


图 4–19　取单元格的当前区域地址

假设不知道“B 组”的数据保存在何处，要求将 B 组所有数据加粗显示，那么可以采用以下代码实现：

```
Sub 加粗显示 B 组的数据()    '放置位置：模块中
    '在所有单元格（即 Cells）中查找“B 组”
    '然后将查找到的单元格的当前区域设置为字体加粗
    Cells.Find("B 组").CurrentRegion.Font.Bold = True
End Sub
```

此过程采用了 Range 对象的 Find 方法查找目标数据，找到后对它的当前区域的字体设置为加粗显示。

代码中 Cells 表示活动工作表的所有单元格，本例代码的含义是在所有单元格中查找目标，查找到的结果是一个 Range 对象，所以可以直接对 Find 的查找结果添加 CurrentRegion 来引用该单元格的当前区域。

如果工作表中所有非空单元格之间没有整行或整列的间隔，那么工作表的 UsedRange 将等同于该区域中任意单元格的 CurrentRegion。

本例案例文件请参考：..\ 第 4 章\ 4-8 CurrentRegion 的应用.xlsm

4.3.9　按条件引用区域：SpecialCells

Excel 提供了一个定位对话框，允许随意指定定位条件，从而选择对应的单元格对象。

按【Ctrl+G】组合键或者【F5】键可以调出如图 4-20 所示的对话框，单击左下角的定位条件可以打开如图 4-21 所示的对话框。在“定位条件”对话框中可以指定“批注”、“常量”、“对

象”等条件，从而选择符合条件的单元格。

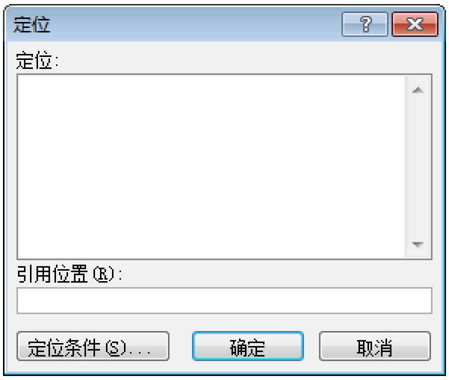


图 4–20 “定位”对话框

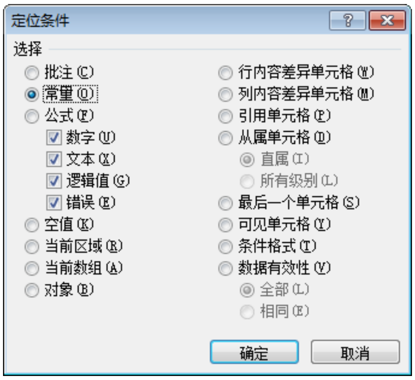


图 4–21 “定位条件”对话框

以上定位条件中使用较频繁的如下。

批注：选择所有包含批注的单元格。

常量：选择所有包含常量的单元格，即除公式外的所有非空单元格。常量中包括数字、文本、逻辑值和错误值。

公式：选择所有包含公式的单元格，同时还可以细分为结果是数字、文本、逻辑值和错误值的公式。

空值：选择所有空白单元格，不包括公式计算结果为空文本的单元格。

当前区域：即 CurrentRegion。

当前数组：即 CurrentArray。

对象：选择所有对象，包含图形对象、图表、文本框和艺术字等。

最后一个单元格：选择已用数据区域右下角的单元格。

可见单元格：选择已用数据区域中非隐藏部分的单元格。

VBA 中的 Range.SpecialCells 方法可以实现以上所有功能，通过 Range.SpecialCells 方法的参数决定定位对象的类型，其语法如下：

```
Range.SpecialCells(Type, Value)
```

其中第一参数 Type 表示定位条件，表 4-1 中包含了可用于定位条件的常量名称及含义。

表 4-1 Type的可选常量名称及其含义

常量名称	含　义
xlCellTypeAllFormatConditions	任意格式单元格
xlCellTypeAllValidation	含有验证条件的单元格
xlCellTypeBlanks	空单元格
xlCellTypeComments	含有注释的单元格
xlCellTypeConstants	含有常量的单元格
xlCellTypeFormulas	含有公式的单元格
xlCellTypeLastCell	已用区域中的最后一个单元格
xlCellTypeSameFormatConditions	含有相同格式的单元格
xlCellTypeSameValidation	含有相同验证条件的单元格
xlCellTypeVisible	所有可见单元格

第二参数 Value 是可选参数，它表示值的类型，包含错误值、逻辑值、数字和文本。只有 Type 参数使用 xlCellTypeConstants 或 xlCellTypeFormulas 时才需要使用 Value 参数。参数的可选常量名称及含义如表 4-2 所示。

表 4-2　Value的可选常量名称及含义

常量名称	值	含　义
xlErrors	16	错误值
xlLogical	4	逻辑值
xlNumbers	1	数字
xlTextValues	2	文本

Value 参数允许多个值相加，例如需要定位活动工作表的所有常量中的逻辑值、数字和文本，那么可以采用 7（1+2+4），完整代码如下：

```
ActiveSheet.UsedRange.SpecialCells(xlCellTypeConstants, 7).Select
```

如果需要定位活动工作表的计算结果为逻辑值与错误值的公式，那么可以采用 20（16+4），完整代码如下：

```
ActiveSheet.UsedRange.SpecialCells(xlCellTypeFormulas, 20).Select
```

Range.SpecialCells 方法在实际工作中相当有潜力，它的优势在于不需要循环，一次性定位所有目标对象，从而避免使用循环语句，提升代码执行效率。

假设工作表中有如图 4-22 所示的数据，要求删除其中所有空行，那么可以采用以下代码：

```
Sub 删除空行() '放置位置：模块中
    '引用活动工作表的已用数据区域
    With ActiveSheet.UsedRange
     '引用工作表最后一列中对应于已用数据区域起止行的单元格
     '也就是说辅助区域在最后一列，其起始行与结束行的行号由 UsedRange 区域的起始行与结束行决定
        With Cells(.Row, Columns.Count).Resize(.Rows.Count, 1)
       '在辅助区域中写入公式，公式的含义是计算已用数据区域中的当前行的数据个数，然后用 0 除以数
'据个数
       '目的是数据个数大于 0（即非空行）时返回 0 值，而数据个数等于 0（即空行）时返回错误值
            .Formula = "=0/counta(" & ActiveSheet.UsedRange.Cells(1).Resize(1, 
ActiveSheet.UsedRange.Columns.Count).Address(0, 0) & ")"
            '在辅助区域中定位结果为错误值的公式所在单元格，然后整行删除
            .SpecialCells(xlCellTypeFormulas, 16).EntireRow.Delete
            '删除辅助区域
            .EntireColumn.Delete
        End With
    End With
End Sub
```

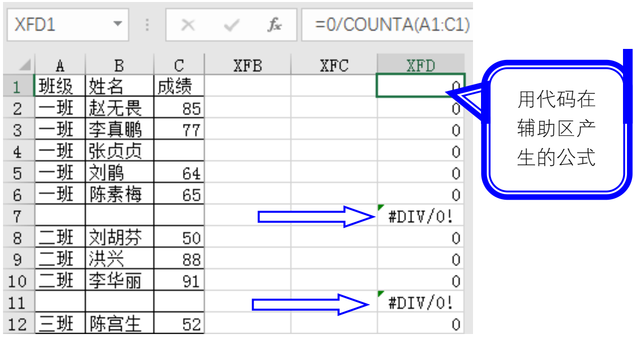
此过程的巧妙之处在于辅助列的应用思路。首先根据活动工作表的已用数据区域的起始行和结束行决定辅助区域的范围，然后在工作表最后一列中创建辅助区域，在其中写入公式，公式的功能是如果当前行有数据则返回零值，否则返回错误值。接着利用 SpecialCells 方法定位公式中的错误值，并将其整行删除，最后删除辅助区域即可。

可以按【F8】键逐步运行代码，查看每行代码的执行结果，从而理解代码的思路。例如当执行到第三句时，可以看到辅助列中的公式如图 4-23 所示。



	A	B	C
1	班级	姓名	成绩
2	一班	赵无畏	85
3	一班	李真鹏	77
4	一班	张贞贞	
5	一班	刘鹃	64
6	一班	陈素梅	65
7			
8	二班	刘胡芬	50
9	二班	洪兴	88
10	二班	李华丽	91
11			
12	三班	陈宫生	52

图 4–22 成绩表



	A	B	C	XFB	XFC	XFD
1	班级	姓名	成绩			0
2	一班	赵无畏	85			0
3	一班	李真鹏	77			0
4	一班	张贞贞				0
5	一班	刘鹃	64			0
6	一班	陈素梅	65			0
7						#DIV/0!
8	二班	刘胡芬	50			0
9	二班	洪兴	88			0
10	二班	李华丽	91			0
11						#DIV/0!
12	三班	陈宫生	52			0

图 4–23 查看辅助区的公式

从图 4-23 的公式可以看出空行与错误值的对应关系。当使用代码产生公式后，只要删除结果为错误值的公式所在的行即可。所以本例的重点在于自己创建条件，为实现目的提供方便。

代码中“Cells(.Row, Columns.Count)”表示辅助区域的第一个单元格，其行号由已用区域的第一行决定，列号是工作表的最大列数，即最后一列。在此代码之后加上“Resize(.Rows.Count, 1)”，表示重置辅助区域的高度，其高度等同于已用数据区域的高度。

本例中最难理解的当属辅助区域的公式，即代码“=0/counta(" & ActiveSheet.UsedRange. Cells(1).Resize(1, ActiveSheet.UsedRange.Columns.Count).Address(0, 0) & ")”的设计思路。

要了解公式的思路首先要查看图 4-23 中的公式与工作表中的已用数据区域的对应关系。以 XFD1 的公式为例，公式为“=0/COUNTA(A1:C1)”，计算结果为 0，公式中的引用区域 A1：C1 刚好对应 ActiveSheet.UsedRange 的第一行，所以可以如此理解：公式中需要调用公式所在单元格对应于已用区域的当前行地址，所以在公式中通过“ActiveSheet.UsedRange.Cells(1).Resize(1, ActiveSheet.UsedRange.Columns.Count).Address(0, 0)”获取该地址，本例中它的计算结果为“A1:C1”。其中 Address 属性用于获取 Range 对象的地址，参数使用两个 0 表示产生相对引用的地址。

本例中 With 的用法比较独特，两个 With 嵌套，外层 With 引用工作表的已用区域，里层 With 引用辅助区域，但是在里层的 With 之内又需要再次引用工作表的已用区域，但此时已不能省略对象，必须完整地书写引用对象名称。

关于变量和变量的数据类型等相关内容请参考本书第 5 章。

本例案例文件请参考：..\ 第 4 章\ 4-9 删除空行.xlsm

4.3.10 模拟 End+方向键产生的单元格：End

Range 对象的 End 属性可以引用新的单元格对象，该对象代表包含源区域的区域尾端的单元格，等同于按组合键【End+向上键】、【End+向下键】、【End+向左键】或【End+向右键】。

例如在图 4-24 中的 B2:F10 区域有数据，活动单元格是 D7，那么使用组合键【End+向上键】可以定位到 D4，使用组合键【End+向下键】可以定位到 D10，使用组合键【End+向左键】可以定位到 B7，使用组合键【End+向右键】可以定位到 F7。

假设在第 12 行和第 Z 列录入新的数据，然后再执行以上操作将得到同样结果。

这是【End+方向键】组合的常规用法，表示在一个区域中根据方向键定位最末端的非空单元格。当工作表中有多个不相邻的区域时，End 仅定位当前区域的边缘。所以如果需要查找某列最后一个非空单元格时需要采用不同的手法。

以图 4-25 为例，不管选择 C1、C2 还是 C5、C8 单元格后再使用【End+向下键】组合键都无法定位到 C 列最后一个非空单元格，所以可以改变操作方式，选择 C1048576 单元格后再使用【End+向上键】组合键定位 C 列最后一个非空单元格。



图 4-24 单数据区域

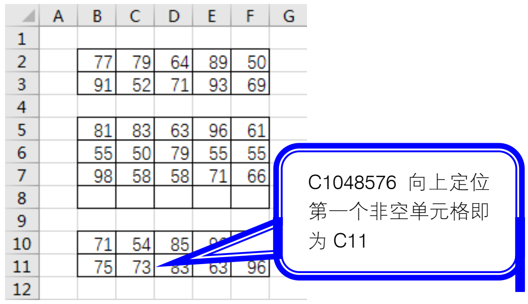


图 4-25 多数据区域

在 VBA 中，Range.End 属性和工作表中的【End+方向键】组合的功能一致。Range.End 属性的语法如下：

```
Range.End(Direction)
```

其中参数 Direction 表示定位的方向，有 4 个可选值，如表 4-3 所示。

表 4-3　Direction参数说明

参　　数	常　　量	说　　明
xlToLeft	-4159	向左
xlToRight	-4161	向右
xlUp	-4162	向上
xlDown	-4121	向下

表中的参数和常量其实具有相同作用，例如以下两句代码都有相同的功能，都表示获取第 1 行最后一个非空单元格的地址：

```
MsgBox Range("XFD1").End(xlToLeft).Address
MsgBox Range("XFD1").End(-4159).Address
```

以上代码中的 Range("XFD1")表示第 1 行最后一个单元格，End(xlToLeft)表示向左定位第一个非空单元格，Address 表示单元格的地址，因此当 Range("XFD1")单元格是空单元格时代码的执行结果为第 1 行最后一个非空单元格的地址，如图 4-26 所示。

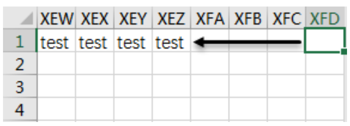


图 4-26　Cells(1,Columns.count).End(xlToLeft)定位示意图

同理，如果需要定位第 3 列最后一个非空单元格，可以采用以下两种方式：

```
Range("C1048576").End(xlUp)
Range("C1048576").End(-4162)
```

以上两句代码在 Excel 2003 中都无效，因为 Excel 2003 只有 65536 行，不支持该代码 Range("C1048576")。在 Excel 2003 中通常用以下代码：

```
Range("C65536").End(xlUp)
Range("C65536").End(-4162)
```

显然，如果不知道用户使用了何种版本的 Excel 时，利用 If 语句判断版本号再决定采用哪一种引用方式不太人性化，因此笔者推荐大家使用以下通用的手法。

定位第 1 行最后 1 个非空单元格：

```
Cells(1,Columns.count).End(xlToLeft)
```

在此代码中 Columns.count 表示列数，它会根据用户的 Excel 版本产生相应的变化，在 Excel 2003 中执行代码结果为 256 列，在 Excel2007 或者更高版本中则为 16384 列。如果在 Excel 2007 和更高版本的兼容模式下执行代码则也会得到 256 列。

Cells(1,Columns.count)表示第 1 行最后一个单元格。当此单元格空白时，从此处向左定位首个非空单元格，就相当于第 1 行最后一个非空单元格。

图 4-26 中的 XFD1 单元格即为 Cells(1,Columns.count)，而 Cells(1,Columns.count).End(xlToLeft)则对应 XEZ1 单元格。

所以以上代码的重点在于 Columns.count，它能自动判断工作表的列数，从而适应版本变化，使代码通用于所有版本的 Excel。

如果要定位第 2 列最后 1 个非空单元格可采用以下代码：

```
Cells(Rows.count,2).End(xlup)
```

此代码中 Rows.count 表示工作表的总行数，Cells(Rows.count,2)则表示第 2 列的最后 1 个单元格，从此处向上定位首个非空单元格就相当于第 2 行最后 1 个非空单元格。图 4-27 为 Cells(Rows.count,2).End(xlup)的定位示意图。

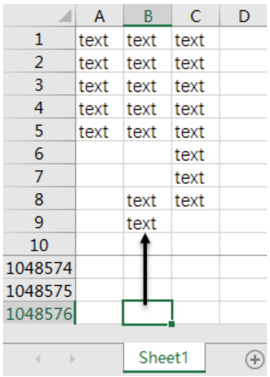
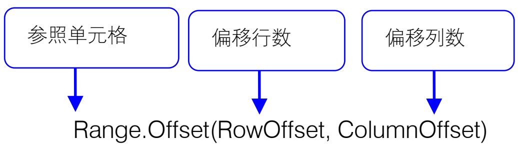


图 4-27　Cells(Rows.count,2).End(xlup)定位示意图

4.3.11　按偏移量重置区域引用：Offset

Range.Offset 是按偏移量引用单元格或者区域，和工作表函数 Offset 的功能与用法都大不相同。此处所言的 Offset 是 Range 的属性，但它的引用结果是 Range 对象。

Range.Offset 的语法如下：



其中 RowOffset 参数表示行偏移量（偏移行数），ColumnOffset 参数表示列偏移量（偏移列数），而前面的 Range 则表示 Offset 的参照对象。简言之，Range.Offset(RowOffset, ColumnOffset)表示相对于 Range 对象偏移 RowOffset 行、偏移 ColumnOffset 列的新区域。

例如相对于 A2 单元格偏移 2 行 3 列的单元格，可用以下代码表示：

```
Range("A2").Offset(2, 3)
```

Range("A2")也可以改用区域，那么通过 Offset 属性生成的新对象也是区域。例如相对于 A2:B3 区域偏移 2 行 3 列的区域，可采用以下代码：

```
Range("A2:B3").Offset(2, 3)
```

此代码引用的对象是 D4:E5，和 A2:B3 区域的高度和宽度一致。所以在使用 Offset 时需要注意参数对象的大小，避免需要引用区域时，Range 对象却是单个单元格，从而不符合预期效果。

Offset 的参数也支持负数和 0，当第一参数使用负数时表示向上偏移，当第二参数使用负数时表示向左偏移。例如 F6 单元格左 2 列上 3 行的单元格可以采用以下代码：

```
Range("F6").Offset(-3, -2)
```

以上引用表示 D3 单元格。如果将第二参数修改为 0，那么表示 F3 单元格，即纵向偏移-3，横向不变。

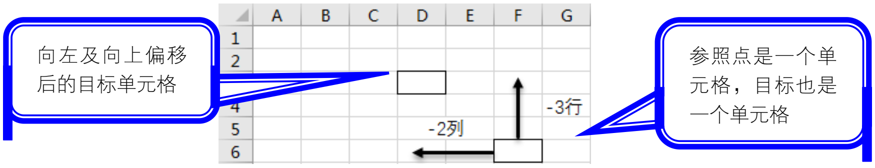


图 4–28　Range("f6").Offset(–3, –2)定位示意图

在实际工作中，Offset 的参数更多的是使用变量，根据某条件计算得到偏移量，从而使代码比以上所举实例更复杂，但思路是不变的，仅仅步骤和代码长度不同罢了。

Range.Offset 引用单元格失败时将弹出提示框“应用程序定义或对象定义错误”。

Offset 通常配合 End 属性使用，例如合并工作表或者合并工作簿时，将所有工作表的数据复制到汇总表中，要求按先后顺序排列，不能覆盖上一次合并的内容。此时需要定位最后一个非空行的下一行，然后再粘贴数据，而达成此需求需要将二者配合使用。

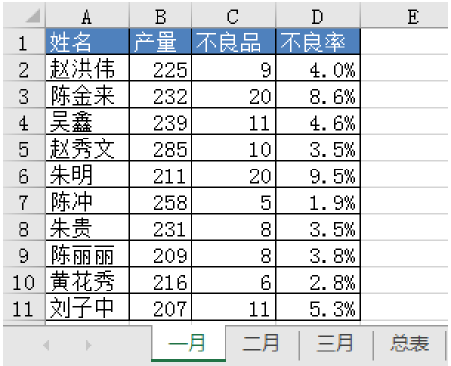
图 4-29 中包括一月、二月、三月和一个空白的总表，要求将三个月的数据合并到汇总表中，那么可以采用以下代码：

```
Sub 合并 3 个月的数据到汇总表() '放置位置：模块中
    '复制一月的已用数据区域到汇总表的 A1
    Worksheets("一月").UsedRange.Copy Worksheets("总表").[a1]
    '复制二月的已用数据区域到汇总表 A 列的最后一个非空单元格的下方
    Worksheets("二月").UsedRange.Copy Worksheets("总表").Cells(Rows.Count, 1).End(xlUp).Offset(1, 0)
    '复制三月的已用数据区域到总表 A 列的最后一个非空单元格的下方
    Worksheets("三月").UsedRange.Copy Worksheets("总表").Cells(Rows.Count, 1).End(xlUp).Offset(1, 0)
End Sub
```

此代码中第一次执行复制时，由于总表是空白表，所以 Range.Copy 方法的参数采用 A1 单元格即可，而第二次和第三次复制时，由于不确定粘贴数据时的目标单元格的位置，所以采用“Worksheets("总表").Cells(Rows.Count, 1).End(xlUp).Offset(1, 0)”计算得来。图 4-30 是合并结果。

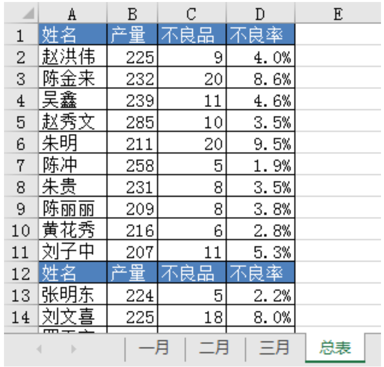
本例中 Range.End 与 Range.Offset 的配合应用很精彩，可以自动找到目标单元格，避免合并工作表后复制的数据覆盖前面的数据。

需要特别强调的是，Range.Offset 的参照点是合并单元格，执行行偏移时将合并单元格中的同一行当做一个整体处理，执行列偏移时将合并单元格中的同一列当做一个整体来处理。



	A	B	C	D	E
1	姓名	产量	不良品	不良率	
2	赵洪伟	225	9	4.0%	
3	陈金来	232	20	8.6%	
4	吴鑫	239	11	4.6%	
5	赵秀文	285	10	3.5%	
6	朱明	211	20	9.5%	
7	陈冲	258	5	1.9%	
8	朱贵	231	8	3.5%	
9	陈丽丽	209	8	3.8%	
10	黄花秀	216	6	2.8%	
11	刘子中	207	11	5.3%	

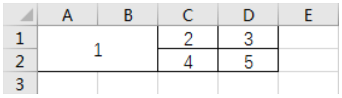
图 4–29 待合并的生产表



	A	B	C	D	E
1	姓名	产量	不良品	不良率	
2	赵洪伟	225	9	4.0%	
3	陈金来	232	20	8.6%	
4	吴鑫	239	11	4.6%	
5	赵秀文	285	10	3.5%	
6	朱明	211	20	9.5%	
7	陈冲	258	5	1.9%	
8	朱贵	231	8	3.5%	
9	陈丽丽	209	8	3.8%	
10	黄花秀	216	6	2.8%	
11	刘子中	207	11	5.3%	
12	姓名	产量	不良品	不良率	
13	张明东	224	5	2.2%	
14	刘文喜	225	18	8.0%	

图 4–30 合并结果

例如图 4-31 中 A1:B2 区域已合并，那么执行列偏移时，参照点用 A1 和 B1 没有区别。



	A	B	C	D	E
1	1		2	3	
2			4	5	
3					

图 4–31 合并单元格

例如以下两句代码的执行结果一致，都是$D$1：

```
MsgBox Range("A1").Offset(0, 2).Address
MsgBox Range("B1").Offset(0, 2).Address
```

当执行行偏移时，参照点用 A1 和 A2 也没有分别。例如以下两句代码的执行结果一致，都是$A$4：

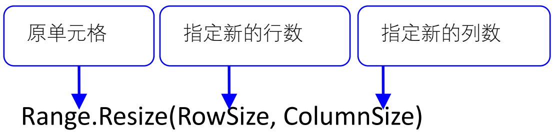
```
Msgbox Range("A1").offset(2,0).Address
Msgbox Range("A2").offset(2,0).Address
```

本例案例文件请参考：..\ 第 4 章\ 4-11 Offset 与 End 的套用.xlsm

4.3.12 按宽度与高度重置区域：Resize

Range 的 Resize 属性表示重新指定区域的高度和宽度，从而产生新的区域引用。

Range.Resize 的语法如下：



其中 RowSize 参数表示高度，ColumnSize 参数表示宽度，不可以是负数或者零值。

例如将 B2 单元格重置为 2 行 3 列的区域，即 B2:D3，可采用以下代码：

```
Range("B2").Resize(2, 3)
```

图 4-32 是 Range("B2").Resize(2, 3)的示意图。

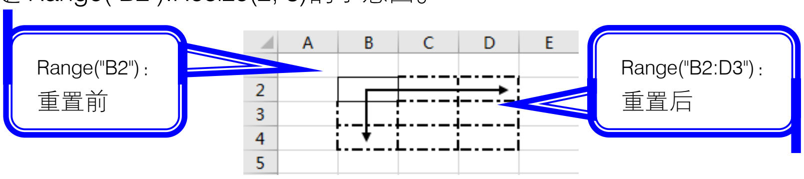


图 4–32 Range("B2").Resize(2, 3)示意图

在实际工作中，Resize 的参数通常采用不确定的值(即变量，在第 5 章会详细介绍)，从而使代码可以根据实际情况的变化自动产生新的引用，适应新的环境，这是 VBA 相对于手动操作的优势。

例如将工作表 Sheet1 的所有数据复制到 Sheet2 工作表中，不保留格式，可采用以下代码：

```
Sub 复制已用区域的值()   '放置位置：模块中
'引用第 1 个工作表的已用数据区域
    With Worksheets(1).UsedRange
      '根据第 1 个工作表的已用区域的宽度和高度定位第 2 个表对应的区域，然后赋值
       Worksheets(2).Range("A1").Resize(.Rows.Count, .Columns.Count) = .Value
    End With
End Sub
```

由于使用 Range.Copy 方法时会将格式一起复制过去，而使用选择性粘贴又无法一个步骤完成，所以本例采用等号将工作表中已用区域的值赋予另一个工作表的相同宽度与高度的区域。

Range.Copy 方法的优势在于只需要指定目标区域的左上角单元格即可，如果本例代码改用 Range.Copy 方法，则可用以下代码：

```
Worksheets(1).UsedRange.Copy Worksheets(2).Range("A1")
```

不过它会将格式信息一并复制过去，不满足当前需求。所以本例采用等号赋值，但需要计算目标区域的宽度与高度。代码中“.Rows.Count”和“.Columns.Count”分别用于计算第一个工作表的已用数据区域的高度和宽度，然后将“Worksheets(2).Range("A1")”对象重置为相同大小的区域，然后执行赋值。

本例中“Worksheets(1).UsedRange”表示第一个工作表的已用数据区域，由于需要引用三次，所以采用 With 语句引用对象，从而简化代码。本例中如果不采用 With 语句，只能如下形式编码（注意此处是一句代码，不换行）：

```
Worksheets(2).Range("A1").Resize(Worksheets(1).UsedRange.Rows.Count,
Worksheets(1).UsedRange.Columns.Count) = .Value
```

4.3.13　引用多区域的合集：Union

Union 是 Application 对象的一个方法，表示将多个单元格对象合并为一个对象，从而在调用对象时更方便。Union 的语法如下：

```
Application.Union(Arg1, Arg2, Arg3, Arg4, Arg5, Arg6, Arg7, Arg8, Arg9, Arg10, Arg11, Arg12, Arg13, Arg14,
Arg15, Arg16, Arg17, Arg18, Arg19, Arg20, Arg21, Arg22, Arg23, Arg24, Arg25, Arg26, Arg27, Arg28, Arg29,
Arg30)
```

Union 的参数至少 2 个，最多 30 个，超过限制时无法合并成功。

Union 通常配合变量使用。换言之，Union 将多个区域合并为一个 Range 对象，然后将它赋值给一个对象变量，以后需要引用这些区域时不再需要引用所有对象，而是调用变量即可，例如：

```
Sub 合并区域()          '放置位置：模块中
    Dim Rng As Range    '声明一个 Range 类型的对象变量
    '将 5 个区域合并为一个对象，然后赋值给变量 Rng
    Set Rng = Union(Range("A1:B10"), Range("c1:c10"), Range("E1:e10"), Range("G1:G10"), Range("i1:i10"))
    MsgBox Rng.Address '获取 Rng 变量的地址，即被合并的 5 个区域的地址
End Sub
```

本例中 Union 将 5 个区域合并为一个对象（注意，不是合并为一个区域，合并后仍然是 5 个区域），然后将它赋值给变量，后续所有需要调用这 5 个区域的地方都可以直接调用对象变量，从而提升代码的编写速度和执行速度。

本例案例文件请参考：..\ 第 4 章\ 4-11 利用 Union 合并区域.xlsm

关于声明变量、变量类型、变量赋值的相关内容请参阅本书第 5 章。

4.3.14 引用多区域的交集：Intersect

Intersect 是 Application 对象的一个方法，表示取多个单元格对象的交集，即重叠部分。Intersect 的语法如下：

```
Application. Intersect (Arg1, Arg2, Arg3, Arg4, Arg5, Arg6, Arg7, Arg8, Arg9, Arg10, Arg11, Arg12, Arg13, Arg14, Arg15, Arg16, Arg17, Arg18, Arg19, Arg20, Arg21, Arg22, Arg23, Arg24, Arg25, Arg26, Arg27, Arg28, Arg29, Arg30)
```

Intersect 的参数至少 2 个，最多 30 个。

取多区域的交集在工作中相当常见，通过以下 3 个案例可以对 Intersect 有较深入的认识。

1．合并工作表

这里仍以图 4-29 的数据为例，要求将三个工作表的数据合并到总表中，但是附加两个要求，其一是标题的行数不确定，其二是合并数据后的总表中只能显示一次标题行。

具体操作步骤如下。

（1）选择菜单“插入”→“模块”；

（2）在模块中录入以下代码：

```
Sub 合并3个月的数据到汇总表()  '放置位置：模块中
    Dim BiaoTi As Byte    '声明一个 Byte 变量，用于存放标题行的数量，标题行一般在 10 行以内
'让用户手动指定标题行的数量
'InputBox 的第三参数 1 表示默认值为 1 行，最后一个参数为 1 表示只允许输入数值
    BiaoTi = Application.InputBox("请输入标题的行数", "确认标题行", 1, , , , , 1)
    '复制一月的已用数据区域到总表的 A1
    Worksheets("一月").UsedRange.Copy Worksheets("总表").[A1]
    With Worksheets("二月").UsedRange    '引用二月工作表的已用区域
        '将已用区域向下偏移标题行，从而产生一个新的区域引用，然后取它与原区域的交集
        '偏移的行数由变量 BiaoTi 决定，而变量的值则由用户手动录入，所以此代码较为灵活
        'Intersect 对偏移前的两个区域获取交集，交集已经排除了标题行，所以复制到总表中时不会有标题
        Intersect(.Offset(BiaoTi, 0), .Offset(0, 0)).Copy Worksheets("总表").Cells(Rows.Count, 
1).End(xlUp).Offset(1, 0)
    End With
    '复制三月的已用数据区域到总表 A 列的最后一个非空单元格的下方（忽略标题行）
    With Worksheets("三月").UsedRange
        Intersect(.Offset(BiaoTi, 0), .Offset(0, 0)).Copy Worksheets("总表").Cells(Rows.Count, 
1).End(xlUp).Offset(1, 0)
    End With
End Sub
```

（3）选择代码后按下【F5】键执行代码，程序会弹出如图 4-33 所示对话框，要求指定标题行数，在对话框中录入数字“1”后单击“确定”按钮，程序会瞬间完成数据合并。图 4-34 是最终合并结果，从图 4-34 中可以确定合并后的总表已经去除多余的标题行。

通过本代码将 3 个工作表的数据合并到总表中，其合并结果比单纯使用 Range.Offset 进行合并的结果要好得多。



图 4-33 手动指定标题的行数

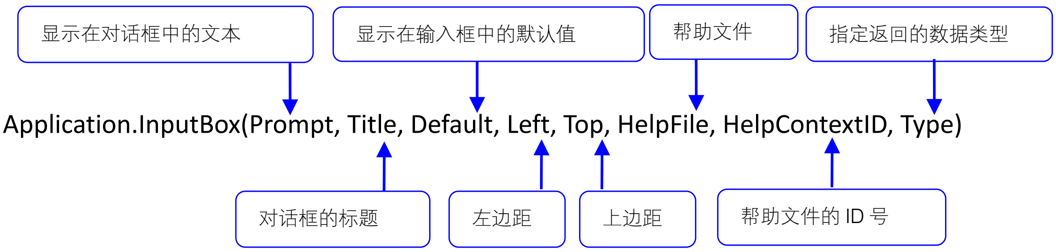


	A	B	C	D
1	姓名	产量	不良品	不良率
2	赵洪伟	225	9	4.0%
3	陈金来	232	20	8.6%
4	吴鑫	239	11	4.6%
5	赵秀文	285	10	3.5%
6	朱明	211	20	9.5%
7	陈冲	258	5	1.9%
8	朱贵	231	8	3.5%
9	陈丽丽	209	8	3.8%
10	黄花秀	216	6	2.8%
11	刘子中	207	11	5.3%
12	张明东	224	5	2.2%
13	刘文喜	225	18	8.0%

一月　二月　三月　总表

图 4-34 合并结果

本例代码的重点有两个，第一个重点是利用 Application.InputBox 方法弹出一个对话框让用户指定标题的行数，Offset 则根据该值执行相应的偏移从而产生新的引用。Application.InputBox 方法的语法如下：



Application.InputBox 的参数名称及说明如表 4-4 所示。

表 4-4　Application.InputBox的参数列表

参数名称	说　明
Prompt	要在对话框中显示的消息。可为字符串、数字、日期或布尔值
Title	输入框的标题。如果省略该参数，默认标题将为Input（中文版为“输入”）
Default	指定一个初始值，该值在对话框最初显示时出现在文本框中。如果省略该参数，文本框将为空
Left	指定对话框相对于屏幕左上角的 X 坐标（以磅为单位），在2010中无效
Top	指定对话框相对于屏幕左上角的 Y 坐标（以磅为单位），在2010中无效
HelpFile	此输入框使用的帮助文件名。如果存在 HelpFile 和 HelpContextID 参数，那么在对话框中将出现一个帮助按钮。通常忽略此参数
HelpContextID	HelpFile 中帮助主题的上下文ID号。通常忽略此参数
Type	指定返回的数据类型。如果省略该参数，对话框将返回文本

只有第一参数是必选参数，其余皆为可选参数。在表格的参数中第一、二、三和第八参数较重要，其他参数通常忽略。本例中采用“Application.InputBox("请输入标题的行数", "确认标题行", 1, , , , , 1)”、就是忽略了第四到第七参数。

参数中最重要的是第八参数，它具有校验功能，可校验输入的值是否符合指定的类型，在表 4-5 中罗列了可用于 Type 参数的常量值及其含义。

表 4-5　Type参数的常量值及其含义

值	含　义
0	公式
1	数字

续表

值	含　义
2	文本（字符串）
4	逻辑值（True 或 False）
8	单元格引用，作为一个 Range 对象
16	错误值，如 #N/A
64	数值数组

从表 4-5 中可以看出来，当 Type 参数为 1 时允许输入数值，当参数为 8 时则限制返回值是区域引用。可以通过以下两句代码测试该参数的实用性：

```
MsgBox Application.InputBox("请输入学号", "只允许数值", , , , , , 1)
MsgBox Application.InputBox("请选择一个区域", "只允许单元格或者区域地址", , , , , , 8).Address
```

执行第一句代码后，在对话框中录入“VBA”，然后单击“确定”按钮，Excel 会弹出如图 4-35 所示的提示框，表示录入的字符未通过校验。

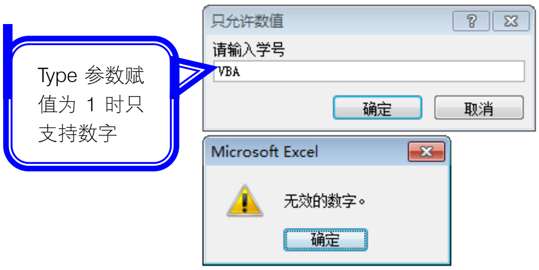


图 4–35　输入文本时无法通过校验

执行第二句代码时 Excel 将弹出一个对话框，此时允许随意选择单元格地址，选择单元格时将在对话框中同步显示所选区域的地址，如图 4-36 所示。

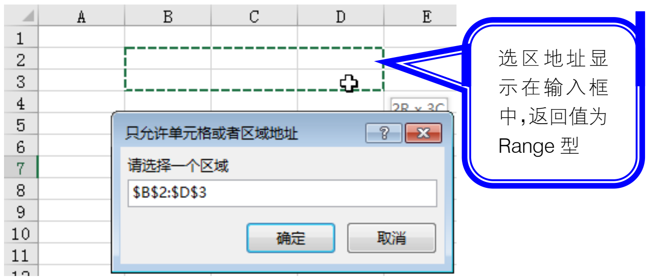


图 4–36　选择区域时在对话框中返回地址

本例代码的第二个重点是 Intersect 与 Offset 的套用，它可以在引用已用数据区域时去除标题行。通过以下简单的代码足以理解 Intersect 与 Offset 套用时的精妙之处：

```
Sub 获取标题行以外的数据区域()          '放置位置：模块中
With ActiveSheet.UsedRange              '引用已用区域
        '获取已用区域与已用区域向下偏移两行得到的新区域的交集
        MsgBox Intersect(.Offset(2, 0), .Offset(0, 0)).Address
    End With
End Sub
```

假设表中有如图 4-37 所示的数据，那么执行以上代码后将弹出图 4-37 标题区以外的数据区域。代码中的 2 表示标题行数。

代码“ActiveSheet.UsedRange.Offset(2, 0)”的执行结果是引用 A3:D10 区域，代码

“ActiveSheet.UsedRange.Offset(0, 0)”的执行结果是引用 A1:D8 区域，所以 Intersect 取两者的交集，即 A3:D8 区域。

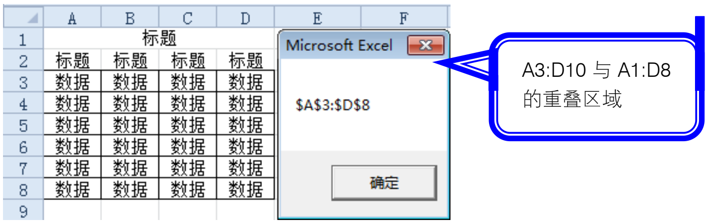


图 4–37　获取标题以外的数据区域

本例案例文件请参考：..\ 第 4 章\ 4-12　用 Intersect 排除标题.xlsm

4.4　图形对象

Excel 的专长虽然不是处理图形对象，但在工作中会经常有处理图形对象的需求，所以如何引用图形对象也是 VBA 的必修课。本节将针对图形对象的多种引用方式逐一分析，在第 7 章中会有关于图形对象的应用案例。

4.4.1　Shapes 对象与子对象

Shapes 代表工作表中的图形对象集合，其父对象是 Worksheet，书写时不可以省略父对象名称。

Shapes 对象包含形状（在 Excel 2003 中称之为自选图形）、剪贴画、SmartArt 对象、图表、艺术字、文本框和插入到工作表中的图片。

Shapes 对象不支持批量操作，即无法对 Shapes 对象执行选中、删除、设置边框等操作，只能通过循环语句对其子对象逐一操作。

在访问 Shapes 的子对象时通常采用序号参数，用法如下。

Shapes(1)——表示引用第一个图形对象。

Shapes.Count——表示图形对象的总数量。

Shapes(Shapes.count)——表示引用最后一个图形对象。

不过在书写 Shapes 对象时不可以忽略父对象，所以即使引用活动工作表中的 Shapes 对象也需要加上 ActiveSheet，否则将产生错误提示“Sub 过程或函数未定义”。例如获取第二个图形对象的名字，必须用以下代码：

```
Msgbox ActiveSheet.Shapes(2).Name
```

4.4.2　图形对象的名称

图形对象有两种命名方式，一种是插入图形对象时在名称栏可以看到的名称，采用汉字加编号的形式命名，可以随意修改；一种是 VBA 中专用的名称，采用英文加编号的形式命名，不可以修改。

通过以下步骤可以了解两种命名方式。

（1）在工作表中插入任意一张图片，在名称栏中可以看到图片名称为“图片 1”，如图 4-38 所示。



图 4–38　名称栏中显示的图片名

（2）按下【Alt+F11】组合键打开 VBE 窗口，然后选择菜单“插入”→“模块”，并在模块中录入代码。

```
Sub Test ( ) '放置位置：模块中
MsgBox ActiveSheet.Shapes(1).Name '获取第一个图形对象的名字
End Sub
```

（3）选择代码，然后按【F5】键执行代码，VBA 将弹出如图 4-39 所示的提示框。



图 4–39　VBA 内部的图片名称

（4）返回工作表界面，进入名称栏，将“图片 1”修改为“图书”。

（5）按【Alt+F11】组合键打开 VBE 窗口，在模块中继续录入代码。

```
Sub 获取图片的边距() '放置位置：模块中
  MsgBox ActiveSheet.Shapes("图书").Left '获取第 1 个图形对象的左边距
End Sub
```

执行代码后将获得图片的左边距，说明手动对图片命名后可以将该名字应用到代码中。但是此方式修改图片名对 VBA 内部的名称没有影响，原名“Picture 1”保持不变。

当工作表中有一个图形对象时，其名称是图形对象的类型加编号 1，当插入第二个图形对象时，则用第二个图形对象的名称加编号 2 命名，不管两个图形对象的类型是否一致。也就是说图形对象的编号与图形对象的类型无关，而是与插入顺序相关。假设表中有两个矩形和两个图表，编号时不会采用“矩形 1”、“矩形 2”和“图表 1”、“图表 2”，而是采用升序的自然数序号，不可能重复，除非手动重新命名。

不过不需要纠结于图形对象的具体名称的写法，在实际工作中，通常采用循环语句访问每个图片，不需要知道图片的具体名称也可以正常工作。

4.4.3　DrawingObjects

前面说过 Shapes 代表工作表中的图形对象集合，但是却不可能通过它对图片执行批量操作，不过 VBA 提供了 DrawingObjects 来弥补此缺陷。

DrawingObjects 是 VBA 的隐藏对象，所以在 WorkSheet 对象的属性列表中无法看到，而且在 VBA 帮助中也没有提及此对象的相关信息。不过此对象在实际工作中为用户带来了诸多便利。

DrawingObjects 对象代表工作表中的所有图形对象，可以批量操作，所以可以通过它实现删除工作表中一切图形对象。通过录制宏可以得到以下完整代码。

```
Sub 宏 1()
    ActiveSheet.DrawingObjects.Select
    Selection.Delete
End Sub
```

以上宏代码中第一句表示选择所有图形对象，第二句表示删除选中的对象。在实际编程时不需要先选择对象再进行删除，直接删除对象即可，所以以上宏代码可以修改为：

```
Sub 删除所有图形对象()  '放置位置：模块中
    ActiveSheet.DrawingObjects.Delete    '删除活动工作表中的所有图形对象
End Sub
```

修改代码后不仅代码更短，执行效率更高，而且减少了选择对象所耗费的时间。

不过，不能对 DrawingObjects 对象集合批量修改属性。例如需要将 DrawingObjects 中的每个子对象与第 2 列对齐显示，即统一图形对象的左边距为 B 列的左边距，用以下代码并不能实现。

```
ActiveSheet.DrawingObjects.Left = Range("B:B").Left
```

DrawingObjects 是一个整体，而 Excel 只允许逐个修改图形对象的属性，所以可以批量删除 DrawingObjects，但在设置属性时需要使用循环语句逐个修改其子对象，请参阅本书 7.2.2 节相关内容。

4.4.4　图形对象的类别子集

图形对象的类型相当多，包括矩形、图表、图片、标注、画布、批注、任意多边形和 SmartArt 图形等。每个类别子集都有一个类别名称，了解这些类别名称在处理图形对象时才更有针对性，例如仅删除图形对象中的图片或者仅调整其中自选图形的 Size 属性。

图形对象的分类方式如表 4-6 所示。

表 4-6　图形对象的类别名称与含义说明

类 别 名 称	值	说　明
msoShapeTypeMixed	-2	混和形状类型
msoAutoShape	1	自选图形
msoCallout	2	标注
msoChart	3	图表
msoComment	4	批注
msoFreeform	5	任意多边形
msoGroup	6	组合
msoEmbeddedOLEObject	7	嵌入的 OLE 对象
msoFormControl	8	窗体控件
msoLine	9	线条
msoLinkedOLEObject	10	链接 OLE 对象
msoLinkedPicture	11	链接图片
msoOLEControlObject	12	OLE 控件对象
msoPicture	13	图片
msoPlaceholder	14	占位符
msoTextEffect	15	文本效果
msoMedia	16	媒体
msoTextBox	17	文本框
msoScriptAnchor	18	脚本定位标记
msoTable	19	表
msoCanvas	20	画布

续表

类 别 名 称	值	说 明
msoDiagram	21	图表
msoInk	22	墨迹
msoInkComment	23	墨迹批注
msoIgxGraphic	24	SmartArt 图形

假设工作表中有图片、图表、矩形、线条等图形对象，要求仅删除工作表的线条，那么可以采用以下代码：

```
Sub 删除直线()              '放置位置：模块中
    Dim i As Integer        '声明一个 Integer 型变量，用于循环语句
    For i = ActiveSheet.Shapes.Count To 1 Step -1   '循环(从图形对象的最后 1 个到第 1 个)
      '如果第 i 个对象的类型是 msoLine（即直线/线条），那么删除第 i 个图形对象
      If ActiveSheet.Shapes(i).Type = msoLine Then ActiveSheet.Shapes(i).Delete
    Next
End Sub
```

代码中图形对象的 Type 属性表示对象的类别，msoLine 表示线条，整个过程的含义是逐个判断所有图形对象的类别名称，如果属于线条就删除，否则忽略。

本例代码中使用了变量和循环的相关知识，本书第 5 章将对变量相关的知识进行详细介绍，而对于循环语句的相关内容请参阅本书 6.2 节。

本例案例文件请参考：..\ 第 4 章\4-13 删除工作表中所有直线.xlsm

4.5 表对象

和单元格对象一样，表对象也是日常工作中会遇到的 Excel 对象，因此了解表对象的特性和引用方式极其重要。

4.5.1 表的合集与子对象

Excel 用 Sheets 表示表对象的集合，Sheets 对象包括工作簿中的所有表。

使用序号可以引用表集合中对应的子对象，示例如下。

Sheets(1)——引用第 1 个表对象。

Sheets(a+5)——引用第 a+5 个表对象，a 是变量。当 a 小于等于–5 时将引用出错。

Sheets.Count——计算表的总数量。

Sheets(Sheets.Count)——引用最后一个表。

Sheets("生产表") ——引用名为生产表的表。

Sheets(Range("B1")) ——引用名字等于 B1 单元格的值的表。

以下代码可以删除最后一个表，但是如果活动工作簿只有一个表时则会操作失败。

```
Sheets(Sheets.Count).Delete
```

4.5.2 表对象的分类

表对象分为五类，具体如表 4-7 所示。

表 4-7 表对象的分类

类型名称	说 明
xlChart	图表
xlDialogSheet	对话框工作表
xlExcel4IntlMacroSheet	Excel 版本 4 国际宏工作表
xlExcel4MacroSheet	Excel 版本 4 宏工作表
xlWorksheet	工作表

其中最常用的是图表 xlChart 和工作表 xlWorksheet，其他三项早已淘汰不用。

VBA 中用 Charts 来表示图表的集合，用 Worksheets 表示工作表的集合。Sheets 对象包含了 Charts 和 Worksheets。例如在图 4-40 中有工作表和图表，通过以下代码可以将它们区分开，同时也能判别 Charts 与 Worksheets 的关系：

```
Sub 计算表的数据()  '放置位置：模块中
    MsgBox "图表数量：" & Charts.Count & Chr(13) & "工作表数量：" & Worksheets.Count
End Sub
```

工作簿中 Chart1 和 Chart2 是图表，其余 3 个是工作表，代码的执行结果如图 4-41 所示。

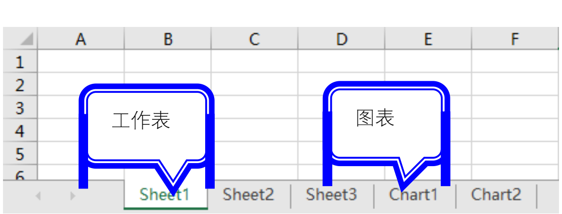


图 4–40 工作表与图表

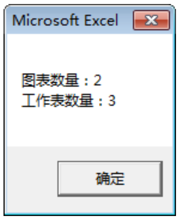


图 4–41 分类统计表的数量

如果将代码改为“Sheets.Count”，那么得到的结果为 5。

如果某个工作表名为“汇总”，那么使用 Sheets("汇总")和 WorkSheets("汇总")都可以正确地引用目标工作表，建议采用后者。

4.5.3 活动表

为了便于区分，VBA 中采用 ActiveSheet 表示活动表，而不用理会当前的活动表是什么类型。也就是说，没有活动工作表和活动图表之分，ActiveSheet 可以代表任意类型的活动表。但是在实际工作中用图表的概率相对于工作表要小很多，因此工作中常将 ActiveSheet 解释为活动工作表。

一个工作簿只允许有一个活动表，如果打开了多个工作簿，那么每个工作簿都有一个活动表。

当访问活动工作簿的活动表时可以忽略工作簿名称，而访问非活动工作簿的活动表时需要指明活动表的父对象。例如将工作簿中“生产.xlsm”的活动表重命名为“二月”，代码如下：

```
Workbooks("生产.xlsm").ActiveSheet.Name = "二月"
```

其中 Name 属性表示表的名称，可以直接赋值从而对表重命名，不过需要遵循 Excel 的规则，例如“\”、“/”、“:”、“?”、“*”等符号不允许作为工作表或图表的名称。

如果要将非活动表变成活动表，那么可以使用 Activate 方法激活它，示例如下。

WorkSheets(4).Activate——激活第 4 个工作表。

WorkSheets(WorkSheets.Count). Activate——激活最后一个工作表。

4.5.4　隐藏工作表的特性

基于各种原因，工作中有时需要将 Excel 的工作表隐藏起来。VBA 可以直接操作隐藏工作表的数据，包括重命名、修改单元格的值、插入行、设置格式等操作。

除取消隐藏工作表的隐藏属性外，不可能手动对隐藏工作表执行其他任意操作，但是 VBA 可以对隐藏工作表执行任意操作。假设第 2 个工作表处于隐藏状态，以下所有语句都可正常执行。

Worksheets(2).Name = "生产表"——对隐藏工作表重命名。

Worksheets(2).Range("B:B").Insert——在隐藏工作表的第 2 列前插入 1 列。

Worksheets(2).Range("B:B").Copy Worksheets(1).Range("A1")——将隐藏工作表的第 2 列数据复制到第 1 个工作表的 A 列。

这其实是 VBA 的特性，即不需要选择对象就可以直接操作。而在手动操作时却需要在选择对象后执行操作，而且无法选择隐藏工作表中的任意对象。

在 VBA 中隐藏工作表有两种方式：一种是普通隐藏，另一种是深度隐藏。

将表的 Visible 属性设置为 xlSheetHidden 即可实现普通隐藏，示例如下。

Sheet2.Visible = xlSheetHidden——将 Sheet2 普通隐藏。

WorkSheets("参数").Visible = xlSheetHidden——将“参数”工作表普通隐藏。

WorkSheets(WorkSheets.Count).Visible = xlSheetHidden——隐藏最后一个工作表。

用代码实现普通隐藏和在工作表界面手动隐藏工作表的效果完全一致。

深度隐藏是指将表的 Visible 属性设置为 xlSheetVeryHidden。当工作表处于深度隐藏状态时，VBA 代码无法对该表执行删除、复制、移动等操作。例如以下代码的第二句将执行失败：

```
Worksheets(2).Visible = xlSheetVeryHidden
Worksheets(2).Delete
```

因此如果工作簿中存在深度隐藏的工作表，那么对其执行批量删除工作表或者批量复制工作表之类的操作之前需要通过代码将工作表还原为显示状态，否则代码将执行出错。

4.5.5　引用名字为数值的工作表的技巧

WorkSheets 代表工作表集合，通过数字参数可以访问集合中的子对象，也可以通过字符串参数访问子对象，那么如果工作表的名字是数值呢？

用数值作为 WorkSheets 的参数时，VBA 会认为该数值是序号，例如 WorkSheets(123)代表第 123 个工作表，而不是名为“123”的工作表。所以如果工作表的名字是数值，那么在引用工作表时需要添加引号。

使用以下代码访问名为“123”的工作表的 A1 单元格，将会弹出如图 4-42 所示的提示信息。错误信息“下标越界”，表示当前代码访问的工作表不存在。

```
MsgBox Worksheets(123).Range("A1").Value
```

正确的代码编写方式如下：

```
MsgBox Worksheets(“123”).Range("A1").Value
```

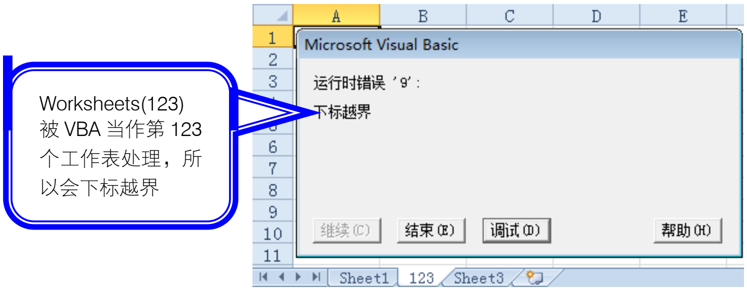


图 4-42　访问 123 工作表的 A1 单元格失败

如果工作表名称来自变量，由于变量名称前后是不允许添加引号的，否则变量名称将变成普通字符串。但是在变量后面添加后缀 “&""” 可以将变量转换成文本，因此当变量是数值时，引用名称来自变量的工作表时可使用以下代码。

```
Sub 取值()              '放置位置：模块中
    Dim a As Byte       '声明一个 Byte 型的变量
    a = 123             '对变量赋值
    MsgBox Worksheets(a & "").Range("A1").Value '取名字为变量值的工作表中 A1 的值
End Sub
```

代码中变量 a 的值是数值 123，但 “a & ""” 却是文本,相当于"123"，所以将它作为 Worksheets 的参数时不再当作序号引用工作表，而是当作工作表名称引用。

4.6　工作簿对象

工作簿就是 Excel 文件，它保存着一切用户数据。本节展示引用工作簿的一些常识和技巧，以及在不同版本的 Excel 中如何确定工作簿的文件格式。

4.6.1　工作簿合集与子对象

VBA 中使用 Workbooks 表示工作簿合集，代表当前已打开的所有工作簿，示例如下。

Workbooks.Count——表示已打开的工作簿数量。

Workbooks.Close——表示关闭打开的所有工作簿。

通过 Workbooks 对象的参数可以引用单个工作簿，示例如下。

Workbooks("工作簿 1")——表示引用新建的工作簿 “工作簿 1”。

Workbooks("财务损益表.xlsm")——表示引用已保存过的工作簿 “财务损益表.xlsm”。

Workbooks(2)——表示引用当前打开的第 2 个工作簿。

Workbooks(Workbooks.count)——表示引用最后一个工作簿。

Workbooks([A1].Value)——表示引用名字等于 A1 单元格的值的工作簿。

4.6.2　活动工作簿

活动工作簿是指当前可以直接操作的处于激活状态的工作簿，与活动表不同，每个工作簿都有一个活动表，所以当前打开多个工作簿时会有多个活动表，但活动工作簿永远只有一个。

VBA 中采用 ActiveWorkbook 表示活动工作簿。

如果需要将一个非活动工作簿转换成活动工作簿，可采用 Activate 方法激活，代码如下：

```
Workbooks("五月.xlsm").Activate
```

引用活动工作簿中的工作表或者单元格时可以忽略 ActiveWorkbook，直接引用表名或者单元格即可。

VBA 中还有一个特殊的对象——ThisWorkbook，ThisWorkbook 表示代码所在工作簿，而活动工作簿指当前正在使用的工作簿，与代码保存位置无关。所以在任何一个打开的工作簿中通过代码调用 ActiveWorkbook 时都指向同一个工作簿，但是在所有打开的工作簿中通过代码调用 ThisWorkbook 时却指向不同的工作簿。

例如当前已打开“生产表.xlsm”和“出货表.xlsm”，活动工作簿是“出货表.xlsm”，那么在两个工作簿的模块中调用 ActiveWorkbook 时，都指向“出货表.xlsm”；但是在“生产表.xlsm”中引用 ThisWorkbook 的名称时只返回“生产表.xlsm”，在“出货表.xlsx”中调用 ThisWorkbook 的名称时只返回“出货表.xlsm”。在实际工作中需要先思考代码操作的对象是什么，然后根据需求选择适合的工作簿引用方式。

4.6.3 关于后缀名

后缀名也被称为文件扩展名，它是操作系统用来标识文件格式的一种机制。后缀名总是跟在主文件名后面，由一个小圆点分隔符分隔开。例如在“abc.def”文件名中，abc 是主文件名，def 是扩展名。主文件名可以随意写，但是扩展名是固定的，用于标识文件格式。

在默认情况下，Windows 隐藏了后缀名，从而给工作带来诸多不便，特别是编程者更需要随时了解当前操作的文件采用了什么后缀名。

为了方便，可以在资源管理器的“文件夹选项”对话框中取消勾选“隐藏已知文件类型的扩展名”，从而使磁盘中所有文件都显示出默认的后缀名。操作界面如图 4-43 所示，调整后的文件显示效果如图 4-44 所示。

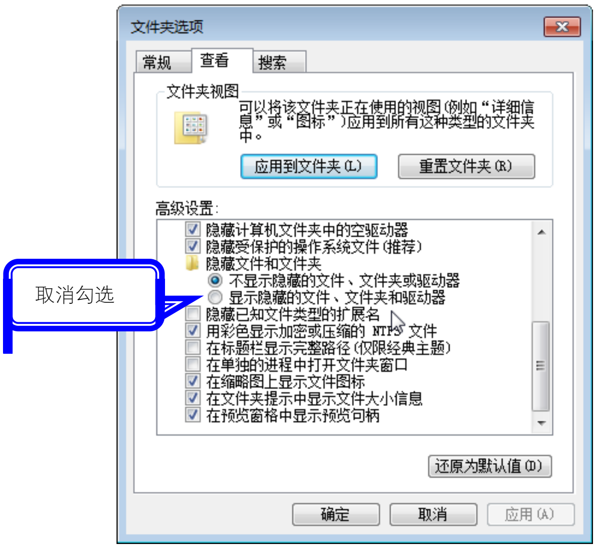


图 4–43　取消隐藏已知文件类型的扩展名

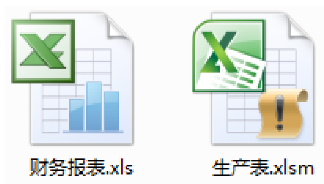


图 4–44　显示文件后缀名

其实取消勾选“隐藏已知文件类型的扩展名”不仅影响文件名的显示效果，还在文件“另存为”对话框中的文件名和保存类型后面也会出现对应的后缀名（见图 4-45），从而给工作带来便利。

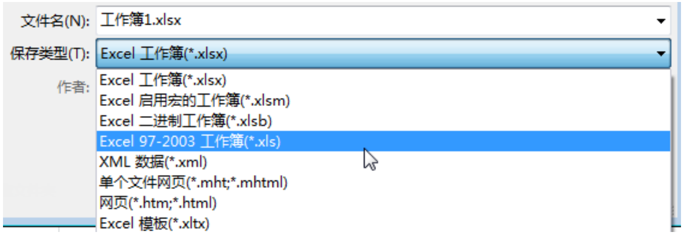


图 4–45　另存为对话框的“保存类型”列表

4.6.4　关于工作簿格式

每个软件的文件都有自己特有的格式，可以通过后缀名进行区分。部分软件还存在不同版本使用不同后缀名的特殊情况。

Excel 2003 及以前的版本皆采用 xls 后缀名，即后缀名为 xls 的文件就是 Excel 工作簿。

从 Excel 2007 开始，微软推行新的压缩格式，且将带宏代码与不带宏代码的文件格式区分开，但是为了确保文件的兼容性，仍然保留了原有格式，所以在 Excel 2007 及更高的版本中通常采用以下三种格式。

◆ xls 格式

这是一种兼容格式，在此格式下不能使用某些新功能，例如最多只能使用 65536 行、256 列，而新版本允许使用 1048576 行、16384 列。在 Excel 2007 及以上版本中，将文件保存为 xls 格式后，如果超过 65536 行、256 列，保存文件时这部分数据将自动消失。但是文件保存为 xls 格式后能确保它的兼容性，在所有版本的 Excel 中都可以正常开启。

◆ xlsx 格式

这是压缩格式，可以使用新版本的一切新功能，而且文件经过了压缩，保存同样多的内容时 xlsx 格式的文件体积远远小于 xls 格式。

xlsx 格式文件可以运行 VBA 代码，但不能保存代码。如果在工作簿中的模块中写入了代码，在保存后会自动丢失代码。

◆ xlsm 格式

这是启用宏的压缩格式，可以使用新版本的一切新功能，也具有文件压缩功能。

xlsm 格式可以保存宏代码，所以学习本书时应尽量将练习文件保存为 xlsm 格式，在不损失功能的前提下，又可以确保代码不丢失。

如果需要将文件转发给 Excel 2003 用户使用，或者不确定代码的使用者使用什么版本的 Excel，那么只能将文件保存为 xls 格式。

此外，Excel 还有 xla 和 xlam 格式的加载宏文件，仅供 VBA 开发者使用，在本书第 14 章将专门介绍制作 xlam 格式的加载宏工具。

4.7　Excel 应用程序对象

Excel 的顶层对象是 Excel 应用程序，采用 Application 表示。在编程过程中会较频繁地调用 Application 对象的属性、方法和子对象。本节简要介绍 Application 对象，并分析不同版本的 Application 对象之间在 VBA 方面的一些差异。

4.7.1 Excel 的顶层对象：Application

Excel 的顶层对象是 Application，Application 有很多子对象。由于顶层对象 Application 只有一个，其书写方法是固定的，因此较其他对象更简单。

以下语句可以关闭 Excel 应用程序：

```
Application.quit
```

有必要将它和“Workbooks.Close”区分开来，“Workbooks.Close”是关闭工作簿，工作簿不等于应用程序，关闭工作簿后 Excel 软件还处于打开状态。所以如果需要关闭所有工作簿后再将 Excel 软件一起关闭，那么只能使用“Application.quit”。

Application 对象有很多属性和方法相当具有实用性，在后面的章节中会有大量应用。

4.7.2 调用子对象时可以省略 Application 吗

Application 有 51 个方法、203 个属性，属性中有几十个属于 Application 的子对象，例如 Rows、Selection、Sheets、ThisCell、ThisWorkbook、Windows、Workbooks 和 Worksheets 等。

在调用 Application 的子对象时可以忽略 Application，直接写子对象名称即可，例如 Application.Selection 简化成 Selection，Application.Workbooks(1)简化成 Workbooks(1)。

不过调用 Application 的方法和子对象以外的属性时，绝大多数情况下不能忽略 Application，只有少数属性可以忽略。建议读者不要忽略 Application 对象，因为编写代码时输入“Application.”会自动出现信息提示，从而可以通过选择目标来录入代码（见图 4-46），这样比手动录入的正确率更高，不需要完整记住所有字母就能输入代码。

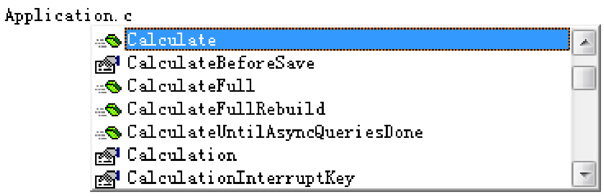


图 4-46 Application 的提示信息

4.7.3 不同版本的 Excel 之间的差异

就目前而言，Excel 的常见版本是 Excel 2003、Excel 2007、Excel 2010、Excel 2013 和 Excel 2016，其中 Excel 2003 接近淘汰的边缘，用户在逐渐转而使用高版本的 office。

Excel 2003 的 VBA 版本是 6.05，Excel 2007 和 Excel 2010 的 VBA 版本是 7.0，Excel 2013 和 Excel 2016 的 VBA 版本是 7.01。所以由于 VBA 的版本不同，Excel 的对象、方法和属性也有微小的差异，因此某些代码注定只能在某个版本中使用，无法通用。

例如 Application.FileSearch 只能在 Excel 2003 中使用，在 Excel 2007、Excel 2010 和 Excel 2013 中都不可以使用；Application.InputBox 的第四参数 Left 和第五参数 Top 仅在 Excel 2003 中生效，而在 Excel 2007、Excel 2010、Excel 2013 和 Excel 2016 都不可以使用；Application.MacroOptions 的第 11 个参数 ArgumentDescriptions 在 Excel 2003 和 Excel 2007 中不允许使用数组，在 Excel 2010 版本开始允许使用数组；

Application.CommandBars.ExecuteMso 只能在 Excel 2007 及以上版本中使用，Excel 2003 中不存在功能区，所以也不可能拥有调用功能区按钮专用的 ExecuteMso 方法；Excel 2010 开始才具有屏幕截图功能，所以以下调用屏幕截图的代码只能在 Excel 2010 以及更高版本中使用。

```
Application.CommandBars.ExecuteMso ("ScreenClipping")
```

Excel 2003 在排序时使用 Range.Sort 方法，从 Excel 2007 开始新增了 Sort 对象，属于 Worksheet 对象的子对象，它的用法和 Excel 2003 的 Sort 方法大不相同。Excel 2003 在排序时可以通过 Sort 方法的 OrderCustom 参数实现按自定义序列排序，而 Excel 2007 开始可以通过 Sort.SortFields.Add 方法的 CustomOrder 参数实现按自定义序列或者按数组排序，在功能上更强大、更灵活。

Excel 2016 和 Excel 2013 有工作簿级的 SheetBeforeDelete 事件，有工作表级的 BeforeDelete 事件，而 Excel 2003、Excel2007 和 Excel2010 没有这两个事件。

尽管如此，也没有必要担心自己开发的程序他人不能用，98%的对象、属性、方法都是一致的，而且如果开发时使用 Excel 2016，而用户也使用 Excel 2007、Excel 2010 或者 Excel 2013 也没有关系，只要尽量不采用 Excel 2016 新增的对象、方法即可。

4.8　课后思考

1．分析 Worksheets(1)和 Worksheets("1")的区别，以及 Activeworkbook 与 Thisworkbook 的区别。

2．同时引用第 6 列、第 5 行的重叠区域和第 2 列、第 5 行的重叠区域（即图 4-47 中写了“目标”的两个单元格），用什么代码?



图 4–47　第 6 列、第 5 行的重叠区域和第 2 列、第 5 行的重叠区域

3．获取第 3 个工作表的 B 列的最后一个非空单元格的地址，用什么代码?

4．用什么代码能实现选择工作表中所有包含公式的单元格?

5．引用“生产表”工作表的已用数据区域中首尾行以外的区域，用什么代码? 例如图 4-48 包含“生产表”中的所有数据，引用其中值为 text 的整个区域。

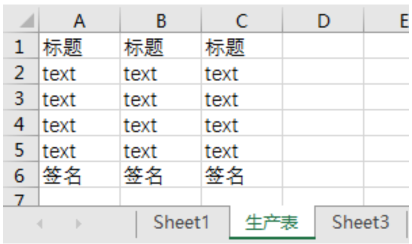


图 4–48　生产表的数据

第 5 章　揭密数据类型与变量、常量

对于 VBA 而言，变量、常量可以大大提升程序的灵活性与效率，让很多不可能实现的事变成可能。但是变量与常量涉及的知识点较多，包括变量与常量的声明方式、数据类型、命名规则、作用域、生命周期等，只有全面深入了解这些基础常识才能发挥常量与变量的强大功能。

本章全面揭密数据类型与变量、常量的概念与应用技巧，为后续学习条件语句与循环语句、常见对象的应用案例等打下坚实的基础。

本章要点

◆ 数据类型

◆ 定义变量

◆ 定义常量

5.1　数据类型

VBA 能处理的数据很多，为了高效地管理这些数据，VBA 中定义了多种数据类型，并将不同的数据分配给不同的数据类型。

一个数据类型就是一类数据的集合，合理地区分数据类型有利于提升程序的执行效率。

5.1.1　区分数据类型的必要性

数据类型的存在价值在于分类管理数据，换言之，数据类型表明了数据是以何种方式储存的，在编程时需要根据实际情况调用合适的数据类型。

每一个数据类型都有一个独有的数据名称，该数据名称表明了数据的有效范围。成绩在 0～59.9 之间表示不及格，而在 60～100 之间表示及格，那么 0～59.9 是数据范围，“不及格”相当于数据名称。

在 VBA 中，系统提供了 Byte、Boolean、Integer、Long、Currency、Decimal、Single、Double、Date、String、Object 和 Variant 等数据类型，每个数据类型都有各自的有效范围。例如 Byte 数据类型表示 0 到 255 之间，Integer 数据类型表示-32768 到 32767 之间。如果当前需要处理的数据是 0 分到 100 分的成绩，那么适合采用 Byte 数据类型，若采用 Integer 数据类型就浪费空间了。在 VBA 中，不同大小的数据在内存中的占用空间是不同的，运算方式也不同，VBA 采用数据类型将它们分类管理。

简言之，数据类型的存在目的就是采用最适合的方式管理数据，从而既能避免空间浪费，又能提高代码的执行效率。

或许采用通俗的比喻更有利于理解数据类型的工作模式和必要性。例如街头有一个自动售水机，张三让 10 岁的儿子张小三使用售水机购水，他需要考虑的问题有两个：

1．由于家里空间有限，所以要根据需求选择不同的盛水容器。例如购 0.1~0.5 升的水时可

以选用 0.5 升的容器；购 0.51 到 1 升的水时统一选用 1 升的容器即可……如果需要 10 到 19 升的水时，那么直接采用可装 19 升的矿泉水桶即可。

为了节约空间，购买 0.2 升水时，就不适合使用 19 升的矿泉水桶。而购买 10 升水时又不能使用 0.5 升的容器，否则容器小于购水量，会造成溢出。

2．需要对容器命名，便于管理和查找。例如将 0.5 升的容器命名为小一号，将 1 升的容器命名为小二号……而将 19 升的矿泉水桶命名为大号。

当以上两个问题解决后，问题就变得简单了。例如要购 0.8 升水，可以直接告诉儿子：儿子，拿小二号打 0.8 升水回来！

如果要购 10 升水，那么直接告诉儿子：儿子，拿大号打 10 升水回来！

当然，有时也可以偷一下懒：儿子，打 0.3 升水回来。至于用什么容器，让儿子自己决定。

在原理上，VBA 中的数据与数据类型和上面的水与盛水容器的关系是一样的。数据类型就是数据的容器，不同大小的数据分配给不同的数据类型，同时将每个数据类型都指定一个名称，例如 Byte 数据类型——有效范围在 0 到 255 之间，Boolean 数据类型——有效范围是所有逻辑值，包含 True 与 False、Integer 数据类型——有效范围在-32768 到 32767 之间……如果将数值 10000 储存在 Byte 数据类型的变量中，那么就会溢出，和小二号容器盛 10 升水一样，而用大号容器盛 0.3 升水就浪费了空间和盛水的效率。

在 VBA 中使用数据类型来整理数据，其目的是优化管理方式，提升工作效率。

5.1.2　数据类型的分类

VBA 提供了 Byte、Boolean、Integer、Long、Currency、Decimal、Single、Double、Date、String、Object 和 Variant 等数据类型，每种数据类型拥有各自独有的名称，以及有效范围，而有效范围决定了它所占用的储存空间。这就像前面所讲的打水容器一样，每个容器都有自己的适应范围，并且对不同的容器采用不同的命名方式。

表 5-1 罗列的 VBA 的常见数据类型和该类数据的有效范围及存储空间大小。

表 5-1　数据类型及其范围说明

数据类型	存储空间大小	有效范围
Byte	1个字节	0 到 255，不支持小数
Boolean	2个字节	True 或 False
Integer	2个字节	-32768 到 32767，不支持小数
Long（长整型）	4个字节	-2147483648 到 2147483647，不支持小数
Single（单精度浮点型）	4个字节	-3.402823E38 到 -1.401298E-45； 正数时从 1.401298E-45 到 3.402823E38
Double（双精度浮点型）	8个字节	负数时从 -1.79769313486231E308 到 484065645841247E-324
Currency（变比整型）	8个字节	从 -922 337 203 685 477 5808 到 922 337 203 685 477.5807，支持小数
Decimal	14个字节	没有小数点时为 +/-79 228162514264337593543950335； 而小数点右边有28位数时为 +/-7.9228162514264337593543950335； 最小的非零值为 +/-0.0000000000000000000000000001
Date	8个字节	100 年1月1日到9999年12月31日
Object	4个字节	任何 Object 引用
String（变长）	10个字节加字符串长度	0 到大约 20 亿

续表

数 据 类 型	存 储 空 间 大 小	有 效 范 围
String（定长）	字符串长度	1 到大约 65 400
Variant（数字）	16个字节	任何数字值，最大可达 Double 的范围
Variant（字符）	22个字节加字符串长度	与变长 String 有相同的范围

需要特别说明的是变体型 Variant，它是默认的数据类型，也是一个全能的数据类型。它能根据数据的大小自动调节自身的类型，从而确保能适应当前的数据。简而言之，需要什么它就变成什么，根据需求自我调节。

这和前面的张三让小儿子自行决定选用什么容器是一样的道理，儿子张小三会根据爸爸交代的购水量自己评估，选用最适合的容器。

变体型 Variant 数据类型的优势可以从其命名看出来，即它的范围会随数据变化而变化，没有固定的有效范围，可应用于所有数据类型。假设待处理的数据是 1000，那么 Variant 将自动把数据当作 Integer 数据类型处理，占用的空间资源与 Integer 数据类型一致；如果数据是 True 则自动当作 Boolean 数据类型处理，变体型的名称正来源于此。

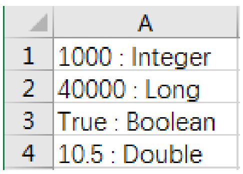
注意：数据类型总是与变量和常量等同时出现，它们是相关联的知识点。关于变量与常量的相关知识将在本章第 2 和第 3 节详细阐述。

通过以下过程可以了解变体型数据类型的变化过程：

```
Sub test()                                    '放置位置：模块中
    Dim a As Variant                          '将变量声明为变体型
    a = 1000                                  '对变量赋值为 1000
    [a1] = a & " : " & TypeName(a)            '将变量的值和变量此时的类型输出到单元格
    a = 40000                                 '对变量赋值为 40000
    [a2] = a & " : " & TypeName(a)            '将变量的值和变量在此时的类型输出到单元格
    a = (1 < 2)                               '对变量赋值为表达式的计算结果，即 True
    [a3] = a & " : " & TypeName(a)            '将变量的值和变量此时的类型输出到单元格
    a = 10.5                                  '对变量赋值为 10.5
    [a4] = a & " : " & TypeName(a)            '将变量的值和变量在此时的类型输出到单元格
End Sub
```

代码中 TypeName 函数用于判断一个数据或者变量的数据类型。Dim 则用于指定变量的数据类型，在本章第 2 节会详细介绍。

上述代码的执行结果如图 5-1 所示。



	A
1	1000 : Integer
2	40000 : Long
3	True : Boolean
4	10.5 : Double

图 5–1　变体型数据类型的特性

图 5-1 的结果说明了对变体型变量赋予不同范围的数据时会根据值的大小自动选择相应的数据类型。对于新手而言，变体型数据类型真是福音，意味着它自动适应数据变化，不用编程者手动指定数据类型。不过虽然如此，仍然有必要手动指定变量的数据类型，原因如下。

第一，VBA 其实并非智能到可以代替人工判断，它有时会分配错误，从而浪费空间。例如当对变体型变量赋值为 1 时，VBA 会将它分配给 Integer 数据类型，而非 Byte 数据类型。参照表

5-1 就可以发现，Integer 数据类型的数据占用的空间是 Byte 数据类型的一倍，那么将一个 1 到 255 之间的整数指定为 Integer 数据类型将会浪费一倍的资源，从而牺牲了程序的效率。这正如张三偶尔偷懒让儿子自行判断，可是儿子可能判断失误一样。

第二，由于每一次赋值 VBA 都会对变体型变量的值执行判断，然后再分配对应的数据类型，所以需要消耗一些时间，从而使程序的执行效率更低。可以通过以下案例证实正确地声明数据类型的重要性。

```
Sub 正确定义变量() '放置位置：模块中
    Dim a As Byte, b As Byte, c As Integer, d As Integer, e As Long, f As Long, g As Double, h As Double, i As Date, j As Date
    i = Timer
    g = 166.666
    For c = 1 To 1000
        For b = 1 To 250
            For a = 1 To 250
                e = CLng((c + b)) * c - b - a - c
                f = CDbl(c) * a / b
                d = c - b + a
            Next a
            g = g + CDbl(c) * b + a / b
        Next b
    Next c
    j = Timer
    MsgBox Format(j - i, "0.00 秒")
End Sub
Sub 不定义变量()  '放置位置：模块中
    i = Timer
    g = 166.666
    For c = 1 To 1000
        For b = 1 To 250
            For a = 1 To 250
                e = CLng((c + b)) * c - b - a - c
                f = CDbl(c) * a / b
                d = c - b + a
            Next a
            g = g + CDbl(c) * b + a / b
        Next b
    Next c
    j = Timer
    MsgBox Format(j - i, "0.00 秒")
End Sub
```

第一段代码对所有变量都正确地定义了数据类型，在笔者的电脑中执行时间为 10 秒左右；第二段代码未对过程的任何变量指定数据类型，所以默认采用变体型 Variant 数据类型，它在笔者电脑中执行时间约为 25 秒钟。由于不同用户的计算机的硬件配置不同，以及操作系统、Office 版本不同，同一段代码在不同电脑中的执行时间会有所差别，但是以上两段代码执行时间的比例基本不会有变化，后者执行时间几乎是前者的两倍。

代码中 Timer 函数表示从午夜零点开始到现在所经过的秒数，带有小数。为了计算代码的运行时间，通常在过程开始时通过 Timer 函数获取当前时间，赋值给变量，然后在过程执行结束时再用 Timer 函数获取代码结束的时间，两个时间相减即为过程的执行时间，Format 函数可将它格式化为单位“秒”，精确到小数后两位。

本例案例文件请参考：..\ 第 5 章\5-1 变体型数据类型的优缺点.xlsm

对于初学者而言，虽然很难记住变量的名称以及变量的有效范围，但是仍然建议读者在编程时尽量对所有变量正确地定义数据类型。在本章下一节中将详细介绍变量与数据类型的关系和应用。

对于表 5-1 中的数据类型名称，其实不需要花费时间记忆，将它打印出来贴在桌上或者夹在笔计本中，也可以直接将该表转成图片保存在手机中，需要时当字典查询即可。个人精力应多放在语法、思路上，而不需要记忆这些固定的知识点。

5.1.3　转换数据类型

前面曾经提到 VBA 会自动根据数据的大小分配数据类型，但有时会分配失误。

有时还存在另一种情况，VBA 不够智能，无法完善、长远地考虑问题。例如：

```
Sub test()'放置位置：模块中
    MsgBox 20000+20000
End Sub
```

执行此过程必定会出错，因为 VBA 会将表达式“20000+20000”中第一个 20000 识别为 Integer 型，同时也将表达式的计算结果预先分配相同的 Integer 数据类型。而事实上该表达式的计算结果为 40000，Integer 的上限是 32767，由于计算结果超过了 Integer 数据类型的上限，因此最终导致执行失败，提示“溢出”。这就和张三让儿子打两瓶 0.4 升的水回来，结果张小三认为 0.4 升就用小一号容器好了，结果一个小一号容器不足以盛两个 0.4 升的水，导致溢出。

再如以下代码中利用工作表的总行数乘以总列数，从而计算工作表中单元格的数量，执行代码时同样提示“溢出”：

```
Sub 计算单元格数量()'放置位置：模块中
   MsgBox Rows.Count * Columns.Count
End Sub
```

错误的根源在于 Rows.Count 的结果是 1048576，VBA 将它识别为 Long 数据类型，而“Rows.Count * Columns.Count”的计算结果 17179869184 超过了 Long 型数据类型的上限 2147483647。

再如以下代码仍然会因为同样的原因而出错：

```
MsgBox 32767 + 1
```

可见 VBA 的自动识别、自动分配是没有前瞻性的，或者说没有后期自我调节功能，它只判断表达式第一个数值，然后自动将结果预先分配为相同的数据类型，当计算结果的值超出预先分配的数据类型的上限时就会提示“溢出”。

所以 VBA 提供了数据类型转换函数用于预防上述问题。常用数据类型转换函数的名称和返回类型如表 5-2 所示。

表 5-2　数据类型转换函数

函　数	返回类型	函　数	返回类型
CBool	Boolean	CLng	Long
CByte	Byte	CLngLng	LongLong
CCur	Currency	CLngPtr	LongPtr

续表

函数	返回类型	函数	返回类型
CDate	Date	CSng	Single
CDbl	Double	CStr	String
CDec	Decimal	CVar	Variant
CInt	Integer		

以上函数具体如何用呢？以 CLng 为例演示，其他函数类似。

前面说过“MsgBox 20000+20000”会执行出错，原因在于 VBA 将计算结果预先设置为表达式中第一个数值的 Integer 数据类型，但实际计算结果超过了 Integer 数据类型的最大范围，所以解决问题的方法是将表达式的第一个值转换成更大的数据类型即可，代码如下：

```
Sub test()'放置位置：模块中
    MsgBox CLng(20000) + 20000
End Sub
```

同理，“计算单元格数量”过程的问题可以采用以下代码解决：

```
Sub 计算单元格数量()'放置位置：模块中
   MsgBox CCur(Rows.Count) * Columns.Count
End Sub
```

CCur 的返回类型是 Currency 型，Currency 数据类型的上限是 922337203685477.5807，所以足够存放“Rows.Count * Columns.Count”的值。

事实上，也可以通过函数来解决以上问题，将数字作为函数的参数参与运算时不存在分配数据类型的问题。以下两句原本会运行时出错：

```
MsgBox 20000 + 20000
MsgBox 2000 * 2000
```

分别改用求和与求积的函数后就不再出错：

```
MsgBox WorksheetFunction.Sum(20000, 20000)
MsgBox WorksheetFunction.Product(2000, 2000)
```

工作表函数是 WorksheetFunction 的子对象，必须添加前缀 WorksheetFunction 才能调用工作表函数。例如在 VBA 中使用“MsgBox Sum([a1],[b2])”无法正常执行，改用“MsgBox WorksheetFunction.Sum([a1],[b2])”才行。

在代码窗口中输入“WorksheetFunction.”后将自动产生工作表函数的名称列表，在列表中选择目标函数即可，这么做既快捷又准确，如图 5-2 所示。

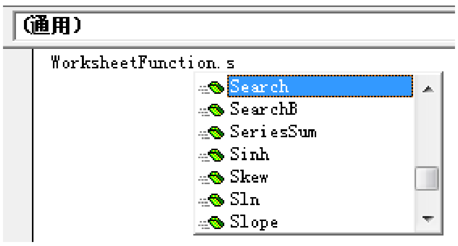


图 5-2 调用工作表函数

值得说明的是，并不需要对以上问题耿耿于怀，虽然执行简单的加法或者乘法时都可能出错，但是实际工作中通常不存在上述问题，而是将数据保存在单元格中，对单元格对象执行计算时不存在分配数据类型的问题。例如在 A1 和 A3 单元格都存放数值 20000，然后执行以下代码不会产生“溢出”错误：

```
MsgBox [a1] +[b3]
```

5.2 定义变量

变量对于 VBA 的重要性不言而喻。没有变量的 VBA 程序只能实现简单的需求，执行很规范的数据运算，而实际工作中有太多不规则的、不可预料的操作对象和操作需求，通过变量可以对一些未知的对象或者数据执行操作，让程序代码有更广阔的应用前景。

5.2.1 变量的用途

变量是指在程序执行过程中随时可能改变其值的量。变量没有固定的值，可以随时根据需求赋予新值。所以变量通常用于代替一个未知数，我们可以在程序中对这个未知数执行操作，而不管这个未知数的值是什么，直到需要时才对未知数赋值或者获取该未知数的值。

下面通过具体的案例来展示。

对 Sheet2 工作表重命名为“生产表”，如果不会编写代码，那么可以采用录制宏的方式实现，宏代码如下：

```
Sub 宏1()    '放置位置：模块中
    Sheets("Sheet1").Select
    Sheets("Sheet1").Name = "生产表"
End Sub
```

第一句代码表示选择工作表，第二句代码表示对工作表命名。可以确定的是第一句代码属于录制宏产生的冗余代码，仅用第二句即可。对所有工作表、单元格、图片、工作簿执行任意操作都不需要选择对象再执行操作。

如果要求产生变化：对 Sheet2 工作表重命名，新名称不确定，由终端用户指定。可见固有的知识不足以实现此类需求，因为工作表的新名称需要采用未知数，而使用变量才是唯一的途径。

我们来看下面的代码：

```
Sub 重命名()  '放置位置：模块中
    '定义一个变量，数据类型为 String（文本都用 String）
    Dim ShtName As String
    '弹出一个对话框让用户指定工作表的新名称，对话框的结果赋予变量
    ShtName = Application.InputBox("请输入新名称", "命名", , , , , , 2)
    '如果输入的文本长度大于 0(即不是空值)，那么用该文本对工作表进行命名
    If Len(ShtName) > 0 Then Sheets("Sheet1").Name = ShtName
End Sub
```

当执行以上过程时，VBA 会弹出一个如图 5-3 所示的对话框，供用户录入字符串，然后使用条件语句判断用户是否输入了文本，如果输入了文本则用它对工作表命名。



图 5-3 可以录入字符串的对话框

在本例中用于命名的字符串是未知的，由终端用户指定，因此编程时谁也无法预料。而变量相当于一个占位符，先用它占位，待变量赋值后自动调用对变量所赋的值参与运算。

调用一切未知的值都需要用到变量，因此在 VBA 中变量的作用极其重要。

本例案例文件请参考：..\ 第 5 章\ 5-2 使用变量对工作表命名.xlsm

此外变量还有简化录入工作、提升程序效率的作用。简化录入是指利用简短的变量代替过长的字符串或者变量名称，给录入工作提供便利；提升程序效率则是指变量储存在内存中，代码调用内存的速度远远高于调用单元格的数据。通过以下两个过程的比较可以印证此观点：

```
'利用变量提升程序执行效率
'第一个过程将 Worksheets("2012 年 12 月生产表").UsedRange 赋予变量，然后用变量参与运算
'由于变量诸存在内存中，读取速度远远快于读取单元格的速度，所以第二个过程执行时间是第一个过程的10 倍
'代码中 Timer 函数用于提取当前的时间，从午夜 0 点开始到现在的时间，单位为秒钟，带有小数
Sub 使用变量()                    '2 秒钟左右('放置位置：模块中)
    Dim Rng As Range               '定义一个 Range 型的对象变量，用它来替代已用区域参与运算
   '将工作表的已用区域赋值给变量 Rng
    Set Rng = Worksheets("2012 年 12 月生产表").UsedRange
    tim = Timer                    '获取当前时间，单位为秒
    '反复执行 30 万次，目的是测试执行时间，执行次数越多越方便比较两种执行方式的时间差异
    For i = 1 To 300000
        a = WorksheetFunction.Sum(Rng)     '计算变量 Rng 所代表的区域的合计，将赋值给变量 a
    Next
    MsgBox Timer - tim             '报告代码执行时间，即当前时间减去上一次提取的时间
End Sub
Sub 不使用变量()                  '20 到 25 秒左右
    tim = Timer                    '获取当前时间，单位为秒
    For i = 1 To 300000            '反复执行 30 万次，目的是测试执行时间
    '对已用区域求合计
        a = WorksheetFunction.Sum(Worksheets("2012 年 12 月生产表").UsedRange)
    Next
    MsgBox Timer - tim             '报告代码执行时间
End Sub
```

第一个过程中将 Range 对象“Worksheets("2012 年 12 月生产表").UsedRange”赋值给变量 Rng，然后用变量 Rng 参与运算，由于变量储存在内存中，所以读取速度很快；第二个过程则每次都读取 Range 对象“Worksheets("2012 年 12 月生产表").UsedRange”，所以执行时间是第一个过程的 10 倍。

本例案例文件请参考：..\ 第 5 章\ 5-3 使用变量提升工作效率.xlsm

5.2.2　定义变量的方法

认识到变量的作用后，下一个工作重点是如何定义变量，应该如何给变量指定数据类型。

在 VBA 中有四种声明变量的方式，包括 Public、Private、Dim 和 Static。

Public 和 Private 用于声明模块级动态变量，Dim 用于声明模块级和过程级动态变量，Static 用于声明过程级静态变量。其中 Dim 语句较为常用。

四者的语法相近，现罗列如下：

```
Dim [WithEvents] varname[([subscripts])] [As [New] type] [, [WithEvents] varname[([subscripts])] [As [New] type]]…
Public [WithEvents] varname[([subscripts])] [As [New] type] [,[WithEvents] varname [([subscripts])] [As [New] type]]…
Private [WithEvents] varname[([subscripts])] [As [New] type] [,[WithEvents] varname [([subscripts])] [As [New] type]]…
Static varname[([subscripts])] [As [New] type] [, varname[([subscripts])] [As [New] type]]…
```

其中方括号中的部分是可选的，“…”表示后面可以继续声明更多的变量。

从语法上看，声明变量的语法比较复杂，但实际上很多参数是可以不用理会的，按以下精简后的方式操作即可：

Dim 变量名称 As 数据类型

Public 变量名称 As 数据类型

Private 变量名称 As 数据类型

Static 变量名称 As 数据类型

在 Public、Private、Dim 和 Static 这四种声明变量的方式中，最常用的是 Dim 语句，其他都较少使用，所以本章重点介绍 Dim 的应用，其他三者的使用方式完全一致，区别仅在于四种方式声明的变量的生命周期不同而已。关于变量的生命周期将在后面详细讲述。

按照 Dim 语句的语法，声明一个名为“ABC”且数据类型为 Byte 型的变量可用以下代码：

```
Dim ABC as Byte
```

如果声明一个名为“成绩”数据类型为 Long 的变量则可用以下代码：

```
Dim 成绩 as Long
```

如果声明 A、C、results 三个 Integer 数据类型的变量则可用以下码：

```
Dim A As Integer, C As Integer, Results As Integer
```

英文变量不区分大小写，所以采用大写或者小写随个人爱好即可。不过如果变量是单词时，首字母大写其余字母小写更具有可读性。

如果声明一个名为“名称”的变体型变量，一个名为“成功”的逻辑型变量可用以下代码：

```
Dim 名称, 成功 As Boolean
```

由于变量默认状态就是变体型，所以声明变体型变量时可以忽略“As Variant”部分。

5.2.3 变量的命名规则

声明变量时需要遵循一定的命名规则，否则会声明失败。变量的名称规则如下。

（1）第一个字符必须使用英文字母或者汉字。

（2）不能在名称中使用空格、点号（.）、惊叹号（!）、@、&、$或#等字符。

（3）名称的长度不能超过 255 个字符。

（4）变量名称不能与自定义的 Function 过程或者内置函数的名称相同。如果常量与内置函数同名，那么在调用内置函数时可在函数名称之前加上关联的类型库的名称。例如定义一个名为 Mid 的变量，需要使用“VBA.Mid”来调用内置的 Mid 函数，否则使用 Mid 时 VBA 会将它看作变量 Mid，从而产生混淆，导致代码执行失败。

（5）不能在同一个模块或者过程中定义相同名称的变量，但在不同模块或者过程中允许存在同名的变量。例如在模块中使用一个“Arr”的公有变量，在过程中可以同时再定义一个名为“Arr”的私有变量。

（6）变量名称不能与 VBA 的保留字一致，例如 Dim、Sub 和 End 等。

（7）变量名称是字母时不区分大小写。

（8）在变量名称中间可以使用分隔符来区分多个单词。例如：Color_Count 或者 Sum_byte 等，但不能是空格或者小数点。

以下声明变量的方式皆不符合要求：

```
Dim 1st as Byte
Dim One-2 as LOng
Dim 二班.成绩 as Byte
```

```
Dim End as Single
```

另外需要补充一点的是，对变量命名时有必要取有意义的名称，提升代码的可阅读性。不宜采用 a、an、one、变量、数据等作为变量的名称，这样既不能通过变量名称了解此变量的数据类型，又不能通过变量名称了解它所代表何种对象。

通常对变量命名可以在以下两种方式中任选一种：

◆ 表明数据类型

例如 String 类型的变量可以采用以下方式声明，只看名称就了解它的数据类型：

```
Dim MyStr As String
Dim Str1 As String, Str2 As String
Dim Inte As Integer
```

◆ 表明变量所代表的对象

直接采用英文单词或者汉字描述变量所代表的对象，例如：

```
Dim 成绩 As Byte
Dim 产量 As Integer
Dim 订单 As Single
Dim SheetName As String
```

对于变量名称采用英文还是中文全凭个人习惯，在执行速度上没有区别。除非代码需要给外国用户使用，否则用中文也不会显得不专业。

5.2.4　变量的作用域

变量的作用域是指变量的可调用范围，包括过程级、模块级和工程级三个级别。

变量的作用域由定义的方式和存放位置决定。定义动态变量采用 Public、Private 和 Dim 三种方式，它们之间的差别如表 5-3 所示。

表 5-3　变量的定义方式与作用域

语　句	代码位置	作用域	说　明
Public	模块顶部	模块级变量	所有模块皆可调用
Private	模块顶部	模块级变量	当前模块可以调用
Dim	模块顶部	模块级变量	当前模块可以调用
Dim	过程内部	过程级变量	当前过程可以调用

表 5-3 没有直观性，接下来通过几个操作步骤直观地比较一下三种声明模块级变量的语句的差异，以及了解 Dim 语句声明模块级和过程级变量的区别。

（1）新建工作簿，使用【Alt+F11】组合键打开 VBE 窗口。

（2）选择菜单“插入”→“模块组合键”，从而创建“模块 1”。

（3）在模块顶部分别采用三种声明方式声明三个模块级变量：

```
Public 模块级公有变量_A As String
Private 模块级私有变量_B As String
Dim 模块级私有变量_C As String
```

其中 Public 所声明的变量是公有变量，所以取名时也加上“公有变量”4 个字便于区分。所谓的模块级公有变量是指所有模块都可以调用的变量，而模块级私有变量则只能当前模块才可以调用。

声明模块级变量的语句只能放在模块的顶部。

（4）继续录入以下两个过程的代码，第一个过程用于对变量赋值，第二个过程用于调用变量的值：

```
Sub One()   '放置位置：模块中
    模块级公有变量_A = "Excel"
    模块级私有变量_B = "VBA"
    模块级私有变量_C = "好犀利"
End Sub

Sub Two()   '放置位置：模块中,其中 Chr(13)表示换行
    MsgBox 模块级公有变量_A & Chr(13) & 模块级私有变量_B & Chr(13) & 模块级私有变量_C
End Sub
```

（5）执行过程“One”对变量赋值，然后执行过程“Two”获取变量的值，可以得到如图 5-4 所示结果。

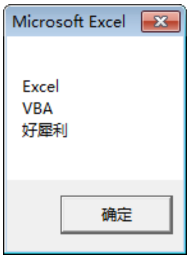


图 5–4　同模块中获取变量的值

从图 5-4 中的结果可以判断：在模块顶部使用 Public、Private 和 Dim 三种方式声明的变量都可以在模块的任何过程中调用，它们都属于模块级的变量。

（6）选择菜单“插入”→“模块”，从而创建“模块 2”。

（7）在模块中录入以下代码，用于再次调用变量的值：

```
Sub Three()   '放置位置：模块中,其中 Chr(13)表示换行
  MsgBox 模块级公有变量_A & Chr(13) & 模块级私有变量_B & Chr(13) & 模块级私有变量_C
End Sub
```

（8）执行过程“Three”，再次调用变量的值，将得到如图 5-5 所示结果，表示只有 Public 所声明的变量可以跨模块调用。



图 5–5　跨模块获取变量的值

接下来，我们继续测试 Dim 语句在声明过程级变量和模块级变量中的区别。

（9）选择菜单“插入”→“模块”，插入“模块 3”，

（10）在模块中录入以下代码，包括声明一个模块级变量和两个 Sub 过程的代码：

```
Dim 模块级变量 As String
Sub Four()   '放置位置：模块中
    Dim 过程级变量 As String
```

```
        模块级变量 = "公有"
        过程级变量 = "私有"
        MsgBox 模块级变量 & Chr(13) & 过程级变量
    End Sub
    Sub Five()
        MsgBox 模块级变量 & Chr(13) & 过程级变量
    End Sub
```

（11）执行过程“Four”对变量赋值，并获取两个变量的值，获取结果如图 5-6 所示。



图 5–6　获取模块级和过程级变量的值

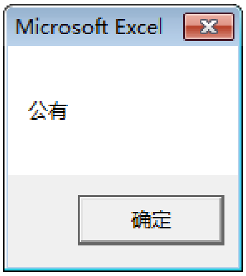


图 5–7　仅成功获取模块级变量的值

（12）执行过程“Five”再次获取两个变量的值，但是只能得到图 5-7 所示的结果。因为变量“过程级变量”只能在声明变量的过程中调用，跨过程调用时只能得到空值。

本例案例文件请参考：..\ 第 5 章\5-4 变量的作用域.xlsm

5.2.5　变量的生命周期

变量的生存周期指对变量赋值后，变量的值在某个时间消失。

让变量的值消失包含人工释放变量和自动释放两种方式，而通常所讲的变量的生命周期是指声明变量后到它自动释放值的时间范围。

变量的生命周期与变量的作用域相关，作用域越大，其生命周期越长。

过程级的变量生命周期最短，当执行完过程后，变量的值就自动消失。前一个案例中过程“Four”声明一个“过程级变量”，当该过程结束后，在过程“Five”中调用时返回空值，这就印证了过程级变量生命周期只在该过程运行的周期之内的说法。

模块级变量的生命周期较长，如果手动删除模块，那么在删除模块时会自动释放变量的值，如果不删除模块，那么只有在关闭工作簿时才释放变量的值。也就是说只要工作簿处于打开状态，模块级变量所占用的内存空间永远都在，随时可以调用。

那么模块级的变量可能手动释放变量吗？答案是可以的，而且有时也是极有必要的。当对多个公有变量赋值后，变量会占用较多的内存空间，释放变量其实就是清空它们所占用的空间，将这些资源投入到新工作中去。就像剪贴板中有太多东西时会影响 Excel 的基本操作一样。

VBA 提供的手动销毁变量的方法是 End 语句，通过以下两个过程可以了解 End 的作用：

```
    Public 模块级公有变量_A As String
    Private 模块级私有变量_B As String
    Dim 模块级私有变量_C As String
    Sub One()  '放置位置：模块中
        模块级公有变量_A = "Excel"
        模块级私有变量_B = "VBA"
        模块级私有变量_C = "好犀利"
```

```
        MsgBox 模块级公有变量_A & Chr(13) & 模块级私有变量_B & Chr(13) & 模块级私有变量_C
        End    '释放一切
    End Sub
    Sub Two()
        MsgBox 模块级公有变量_A & Chr(13) & 模块级私有变量_B & Chr(13) & 模块级私有变量_C
    End Sub
```

首先执行过程“One”，对变量赋值，然后获取变量的值，可以发现三个变量的值都可以获取成功。在过程的末尾处有一个“End”语句，它能释放一切变量，包括过程级变量、模块级私有变量和模块级公有变量。

然后执行过程“Two”再次获取变量的值，发现只能获得空文本，说明变量已被成功释放。

本例案例文件请参考：..\ 第 5 章\5-5 手动释放变量.xlsm

注意：End、End Sub、Exit Sub 的区别如下：

End Sub 语句仅仅用于终止当前过程，是过程结束的标志，不可以放在过程中间。

Exit Sub 语句也表示终止当前过程，不过它可以放在过程中任意位置，可以随心所欲地在任何需要的时候终止过程。

在以下过程中使用了两次 MsgBox 函数，正常可以弹出两次信息提示框，但在中间插入 Exit Sub 语句后，第二句 MsgBox 将会被忽略。

```
    Sub test()             '放置位置：模块中
        MsgBox 123
        Exit Sub           '终止过程
        MsgBox 456
    End Sub
```

End 语句和 Exit Sub 语句的功能相近，都可以在中途终止过程，不过 End 语句多了一个释放所有变量的功能。使用 End 语句后，当前工程中所有变量都被释放掉，所以要根据需求选择用 Exit Sub 语句还是 End 语句。

5.2.6 静态变量与动态变量的区别

前面所讲的所有过程级变量都是动态变量，过程级的动态变量有一个鲜明的特点——过程结束时自动释放变量的值。在工作中，有时需要过程级变量保留变量的值，以便下一次调用，而 VBA 提供了一种新的声明方式——Static 语句。

Static 语句声明的变量属于静态变量，它只能声明过程级变量，所以 Static 语句不能放置在模块顶部。

通过以下过程足以了解静态变量与动态变量的区别：

```
    Sub test()                                        '放置位置：模块中
        Static a As Byte                              '声明一个静态变量
        Dim b As Byte                                 '声明一个动态变量
        a = a + 1                                     '将变量在原值基础上加 1
        b = b + 1                                     '将变量在原值基础上加 1
        MsgBox "a：" & a & Chr(13) & "b：" & b        '获取两个变量的值
    End Sub
```

执行以上过程后可以得到图 5-8 所示结果，此时看不出两个变量有何区别。

当反复执行过程时就可以发现二者本质上是不同的。例如第三次执行时能取得如图 5-9 所示

的结果，这意味着动态变量的值自动消失，而静态变量的值保留了下来。



图 5-8　第一次获取变量的值



图 5-9　第三次获取变量的值

静态变量在工作中有其用武之地，例如以下过程仿制一个录入密码的对话框，如图 5-10 所示。如果录入正确则弹出如图 5-11 所示的提示框，然后执行后续的任意操作；如果录入不正确则弹出对话框提示错误，如果三次录入错误则自动关闭工作簿。

```
Sub 利用静态变量验证录入密码次数()          '放置位置：模块中
    Static Num As Byte                      '声明一个静态变量，用于存放录入密码的次数
    Dim Mima As String                      '声明一个动态变量，用于存放用户录入的密码
    Mima = Application.InputBox("请录入密码:", "密码", , , , , , 2)    '让用户录入文本型密码
    Num = Num + 1                           '记录录入密码的次数
    If Mima = "VBA Good" Then               '如果录入的密码是“VBA Good”，那么提示验证成功
        MsgBox "验证成功", vbOKOnly + vbInformation, "恭喜"
    Else                                    '否则（表示输入有误）
        If Num < 3 Then                     '如果输入次数小于三次
            MsgBox "密码错误，请重新输入"'提示输入错误
        Else                                '否则（表示错误三次）
            MsgBox "对不起,密码三次错误", vbOKOnly + vbInformation, "可惜" '提示用户
            ActiveWorkbook.Close , False    '自动关闭工作簿
        End If
    End If
End Sub
```



图 5-10　“密码”对话框



图 5-11　“验证成功”提示框

本例案例文件请参考：..\ 第 5 章\5-6 静态变量的应用.xlsm

5.2.7　声明对象变量

在 VBA 中除了前面提到的各类数据变量，还有一种很特殊的、应用相当频繁的对象变量。顾名思义，对象变量用于表示对象，只能将对象赋值给对象变量，而不能将字符串或者数值赋值给对象变量。

对象变量的种类相当多。Excel 的每一种对象都对应一种变量类型，例如单元格对象的变量

类型是 Range，工作表对象的变量类型是 WorkSheet，工作簿对象的变量类型是 WorkBook、图形对象的变量类型是 Shape……

声明对象变量和数据变量的方法一致，都是通过 Public、Private、Dim 和 Static 四者之一实现，唯一区别在于变量的类型名称不同。

当录入 As 加空格后，VBA 会自动弹出变量类型的名称提示，只需录入首个字母就会自动定位到该字母开始的单词处。在图 5-12 中，由于 as 语句后面输入了字母 r，所以该提示定位到 r 开头的首个对象类型名称 Range 处。

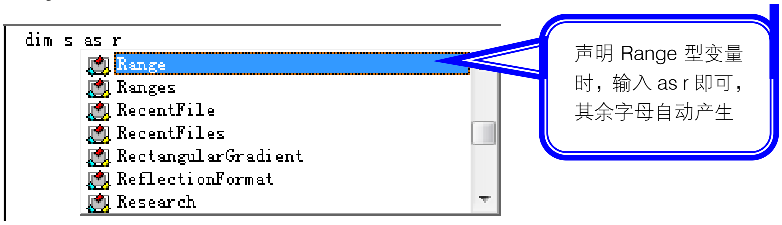


图 5–12　变量类型名称提示

在如图 5-12 所示的列表中，图标为 者表示数据变量名称，图标为 者表示是对象变量名称。常用的对象变量如表 5-4 所示。

表 5-4　常用对象变量名称

对 象 名 称	对 象 变 量 名 称	对 象 名 称	对 象 变 量 名 称
工作簿	WorkBook	批注	Comment
工作表	WorkSheet	名称	Name
单元格	Range	图形对象	Shape
图表	Chart	不确定对象	Object

类似于数据变量中的变体型 Variant 数据类型，对象变量类型中的 Object 也很全能，它可以替代所有对象变量类型，当不确定对象变量的名称时都可以用它替代。

以下是声明对象变量的常规手法：

```
Dim Rng As Range
Dim Cell As Range
Dim Sht As Worksheet
Dim Wb As Workbook
Public Com As Comment
Public 成绩 As Range
```

对数据变量赋值直接使用等号即可，而对对象变量赋值需要使用 Set 语句。例如对 Rng 变量赋值为 A1 单元格，可用以下代码：

```
Set Rng = Range("A1")
```

如果忽略 Set 语句，那么将会运行出错，除非已经用 Set 对对象变量赋过值。

对对象变量赋值需要使用 Set 语句，但是其后对变量所代表的对象赋值时不再使用 Set，例如以下过程：

```
Sub 对象变量()                '放置位置：模块中
    Dim Rng As Range        '声明一个 Range 型的对象变量
    Set Rng = Range("A1")   '将对象 A1 单元格赋值给变量 Rng
    Rng = Range("A2")       '将 A2 单元格的值赋值给 Rng 所代表的单元格（即 A2 的值复制到 A1 中）
End Sub
```

过程第一句表示声明对象变量，第二句表示将 A1 单元格赋值给变量 Rng，此时变量 Rng 就

代表 A1 单元格。

第三句代码“Rng = Range("A2")”其实相当于“Rng = Range("A2").Value”，它的作用和第二句明显不同，它表示将 A2 单元格的值赋值给变量 Rng 代表的单元格对象 A1。此时“Rng = Range("A2")”表示对 Rng 所代表的对象写入数据，而不是对变量 Rng 赋值。

换言之，“Set Rng = Range("A1")”表示将变量 Rng 与 Range("A1")之间建立关联，使 Rng 可以代表 Range("A1")参与各种运算；而代码“Rng = Range("A2")”则是将 Range("A2")的值复制到 Range("A1")中。

读者可以在 A2 单元格录入任意值，然后执行以上代码测试是否实现将 A2 的值复制到 A1 中。

Range 对象的默认属性是 Value，即值属性，除了 Set 语句，如果忽略 Range 对象的属性都表示获取它的值。所以“Rng = Range("A2")”其实相当于“Rng = Range("A2").Value”。

虽然如此，为了区分“Set Rng = Range("A1")”与“Rng = Range("A2")”中 Range 对象变量的作用，在编写代码时尽量将 Value 属性一并书写完整，从而提升代码的可读性。

Set Rng = Range("A1")——此处表示将单元格对象 Range("A2")赋值给变量。

Rng = Range("A2").value——此处表示将单元格对象的值赋值给变量所代表的对象。

5.2.8 对象变量的初始化与释放

对象变量在使用 Set 赋值前属于未初始化的变量，此时虽然也已经分配内存空间，但它还属于 Nothing，表示啥也没有。可以使用 TypeName 函数判断变量是否已经初始化，示例如下。

```
Sub 对象变量()                  '放置位置：模块中
    Dim Rng As Range            '声明 Range 型的对象变量
    MsgBox TypeName(Rng)        '判断变量是否初始化
    Set Rng = Range("A1")       '对对象变量赋值（初始化）
    MsgBox TypeName(Rng)        '再次判断变量是否初始化
End Sub
```

运行此过程后，两个 MsgBox 函数分别得到如图 5-13 和图 5-14 所示的结果，表示声明 Range 变量之后、对变量赋值之前变量属于 Nothing，而使用 Set 语句赋值后变量将成为 Range，此时变量 Rng 代表 A1 单元格，对 Rng 变量的一切操作都会在 A1 单元格中体现出来。



图 5-13 判断结果 1

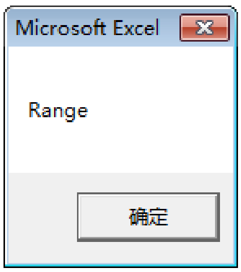


图 5-14 判断结果 2

注意：“Set Rng = Range("A1")”语句不可以写作“Set Rng = Range("A1").value”

判断变量是否初始化还可以使用 Is 运算符，它用于比较两个变量是否相等，通常配合 If 语句判断。例如判断 Rng 对象变量是否初始化可用以下代码：

```
If Rng Is Nothing Then MsgBox "未初始化" Else MsgBox "已初始化"
```

如果需要释放对象变量，即把已初始化的对象变量还原为 Nothing，那么可以使用 Set 语句

对其赋值 Nothing，代码如下：

```
Set Rng=Nothing
```

当然，如果要批量释放所有对象变量的值，那么还是用 End 语句较为快捷。

5.3 定义常量

在 VBA 中，常量和变量的用法完全一样，只是在重要性方面略差于变量，但是善用常量也能提升编程的效率。

5.3.1 常量的用途

常量也称常数，是指执行程序时保持常数值的命名项目，通俗地讲就是指代码执行过程中不变的值。常数可以是字符串、数值以及除乘幂和 Is 之外的算术运算符或逻辑运算符的组合。“中国”、“VBA”、“028” 等都是常量。

常量包括两种，即内部定义的常量和用户通过 Const 语句自定义的常量。

常量主要的功能有如下两个。

◆ 便于识别

在使用公式时常会将某些区域或者数值定义为一个比较有意义的名称，当需要调用这些区域或者数值时可以用名称替代。而 VBA 中的常量就相当于公式中常用的名称。

例如某项产品的生产困难系数标准是 102.236458%，它在一个或者多个过程中会重复出现，那么将它定义为“困难系数”，以后直接用常量“困难系数”参与运算即可，这与用 102.236458% 参与运算的结果完全一致，但是用常量更易于识别，只要看到常量名称就知道它的作用，而在代码中出现 102.236458%时，则需要花费更多的时间才能明白它的含义。

◆ 简化输入

正如前面所介绍的，将 102.236458% 这种较长的字符串定义为常量，在需要使用的地方直接采用常量即可，不需要再录入该字符串，显然常量更便于记忆和录入。

再如将 3.1415926 定义为 pai，输入 pai 显然较输入 3.1415926 方便、快捷，同时还能确保准确性。编程时输入的字符越长，出错的概率越高。

5.3.2 常量的定义方式

定义常量采用 Const 语句，其语法如下：

```
[Public | Private] Const constname [As type] = expression
```

Const 的参数列表如表 5-5 所示。

表 5-5 Const的参数列表

参 数	是否必要	说 明
Public	可选	该关键字用于在模块顶部声明常量，该声明的常量可以被所有模块中的过程所调用。不能在过程中使用此参数
Private	可选	该关键字用于在模块顶部声明常量，该声明的常量只能被当前模块中的过程所调用。不能在过程中使用此参数
constname	必选	常数的名称；遵循标准的变量命名约定
type	可选	常数的数据类型；可以是 Byte、 Boolean 、Integer、Long、Currency、Single、Double、Date、String 或 Variant。所声明的每个变量都要使用一个单独的As类型子句
expression	必选	文字，其他常数，或由除 Is 之外的任意的算术操作符和逻辑操作符所构成的任意组合

简单地说，定义常量采用以下语法：

```
Const 常量名称 As 数据类型 = 常量值
```

而前面是否加 Public、Private 取决于是否打算将此常量用在其他模块，多数情况下是本过程或者本模块使用，忽略该部分即可。

常量的命名方式与变量完全相同，不再赘述。

以下是一些常见的常量定义方式：

```
Const output   As Integer = 1285
Const 地区 As String = "广东省广州市火车站"
Const 系数   As Currency = 124.457654
```

也可以使用日期型常量，例如：

```
Const 日期 As Date = #12/21/2012#
```

注意：声明日期型变量或者常量时，数据类型必须使用 Date。对日期型常量赋值时需要采用“#”，对日期型变量赋值则区分两种形式，如果直接使用常数赋值，则需要添加“#”；如果该值在单元格中，则使用单元格对象即可。例如以下两种变量赋值的方式都是允许的：

```
Dim 日期 1 As Date, 日期 2 As Date
日期 1 = #8/8/2008#
日期 2 = Range("A1").Value
```

以下三种声明常量的方式皆不可取：

```
Const 单价 As Byte = Sheets("单价表").Range("B2").Value
```

由于对常量赋值只能使用常数，而上面的代码中引用了单元格，所以是不允许的。但是可以使用简单的表达式，例如：

```
Const 成绩 As Byte = 65 + 3
```

但是不允许使用函数，Format 是 VBA 的函数，所以以下代码会执行失败：

```
Const 成绩 As Byte = Format(12.538, "0.00")
```

常量的类型只能是数据类型，不能使用对象的类型名称，也不能将对象赋值给常量。所以以下代码也会执行失败：

```
Const Rng As range =ActiveCell
```

5.3.3　变量与常量的异同分析

变量和常量对 VBA 而言都比较重要，其中变量显得尤其重要。它们的概念和使用方式相近，在细节上有些差异，主要体现在以下几点。

（1）声明常量时可直接赋值，而变量只能在声明后另起一行赋值。

（2）常量的值永远不会变化，变量的值可以随时变化。

（3）对常量赋值只能用常数，而对变量赋值则无任何限制，再复杂的表达式都可以。

（4）变量的数据类型可以包括对象和数据，而常量只能是数据。

（5）对变量赋值可以使用函数，对常量赋值不可以使用函数。

5.4　课后思考

1．为以下几个变量指定数据类型，填写在括号中。

99(　　　　)、"上海码头"(　　　　)、True(　　　　)、-23.58(　　　　)、99999(　　　　)、

#21/28/2013#(   )、"459" (   )、128*6(   )

2．如果一个变量“产量”需要在多个模块中调用，需要如何声明这个变量？如果只需要在当前模块中能调用又该如何声明？

3．如何对对象变量赋值？如何释放对象变量的值？

4．下面的 5 句代码出错的原因是什么？

```
Const a As String = Range("A1").Value
Dim Sheet1 As Rnage
Set Rng = Worksheet(1)
Dim STr as String = 112
Const Tim As Date = "#2013/8/9 14:59:80#"
```

5．分析在编程过程中正确定义变量有哪些优势。

第 6 章　条件语句与循环语句

条件语句可以让程序有选择性、有针对性地执行，从而提升程序的灵活性。

循环语句可以让程序批量操作，用简短的代码处理复杂的问题，从而全方位提升程序的执行效率。

条件语句和循环语句是 VBA 基础知识中最重要的一课，本章将详细剖析其语法与应用思路。

本章要点

◆ If 语句解析
◆ Select Case 语句解析
◆ IIf 函数
◆ For Next 语句解析
◆ For Each Next 语句解析
◆ Do Loop 语句解析

6.1　If 语句解析

If 语句是 VBA 中最好用的条件语句，它灵活、简单、易学，同时又易懂，是 VBA 初学者最容易掌握的条件语句。

If 语句有多种写法，也有多种用法，虽然都很简单，但是初学时很容易混淆。

6.1.1　条件语句的重要性

录制宏对于 VBA 新手和老手都极为重要，新手可以通过录制宏直接得到代码，通过“播放代码”一键完成同样的工作，从而解放双手。老手虽然懂得编写代码，但仍然会通过录制宏取得代码，然后再改造、完善代码，使其满足实际需求，这比手动编写代码更快捷、更不易拼错单词，也更节约精力。

不过录制宏只针对普通的操作，条件语句是永远无法录制出来的，所以仅靠录制宏不可能完成有针对性的操作。

删除第 5 行或者在 A5 单元格插入整行这类简单操作，利用前面所学知识足以应对，用以下两句代码即可。

```
Rows("5:5").Delete
Range("A5").EntireRow.Insert
```

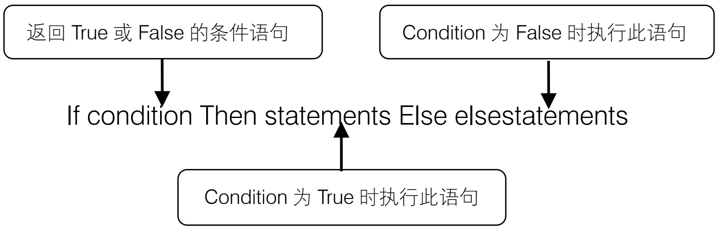
然而在实际工作中，可能需求为“如果第 5 行是空行就删除”或者“如果 A5 是文本则插入 1 行，否则插入 2 行”，对于这类要求必须借助条件语句才能实现。

没有判断语句的代码永远不具备智能，无法满足工作中复杂多变的需求。

6.1.2 If...Then...Else 的单行模式

最简单的条件语句是 If Then 语句，它表示如果符合条件，那么执行一组语句，否则执行另一组语句。

If Then 包含单行模式和块形式。单行模式的基本语法如下：



其中 condition 部分表示条件，通常是运算结果为 True 或者 False 的表达式。

statements 部分表示条件为真时需要执行的语句，可以是一句也可以是多句。

Else elsestatements 部分是可选的，表示不符合条件时需要执行的语句，可以是一句也可以是多句。

假设 A1 单元格存放的是语文成绩，如果 A1 的值大于等于 60 就在 B1 单元格显示“及格”，否则显示“不及格”。按照以上语法将得到以下代码：

```
Sub 判断()  '放置位置：模块中
If Range("A1").Value >= 60 Then Range("B1").Value = "及格" Else Range("B1").Value = "不及格"
End Sub
```

在书写代码时需要注意 Then 和 Else 前后皆有空格，否则代码可能无法执行。

如果要求 B10 单元格非空则在 B10 单元格处插入整行，如果空白则保持不变，可以采用以下代码：

```
If Len(Range("B10").Value) > 0 Then Range("B10").EntireRow.Insert
```

“Len(Range("B10").Value) > 0”表示单元格的文本长度是否大于 0，也就是判断是否空白。此表达式的返回值为 True 或者 False，本例中如果返回 True，则执行代码“Range("B10").EntireRow.Insert”实现整行插入，否则什么也不做。

如果要求最后一个表是“总表”时，将工作表“生产表”的所有数据复制到“总表”中，那么可以使用以下代码：

```
If Sheets(Sheets.Count).Name = "总表" Then WorkSheets("生产表").UsedRange.Copy Sheets("总表").Range("A1")
```

以上三个案例都是符合条件或者不符合条件时执行一句代码，如果需要执行多句代码，有以下两种处理方案。

其一是使用冒号连接多句代码，例如当 A1 小于 100 时，则将 A1 设置为黑体且加粗，代码如下：

```
If Range("A1").Value < 100 Then Range("A1").Font.Name = "黑体": Range("A1").Font.Bold = True
```

代码中的冒号表示将多句代码放在同一行中，按从左到右的顺序执行。

其二是采用 If 语句的块形式，在本书 6.1.5 节详细描述这方面内容。

注意：单行模式的 If 语句没有 End If。

6.1.3　And、Or 和 Not 在条件语句中的作用

And、Or 和 Not 属于三种逻辑运算，常配合条件语句使用。

1．And 运算

And 运算表示对两个表达式进行逻辑连接，如果两个表达式的运算结果都是 True，则返回值为 True，否则为 False。简单地讲就是双条件判断，在两个条件都满足要求时返回 True。

And 的语法如下：

result = expression1 And expression2

其中 result 表示返回值，expression1 和 expression2 分别表示条件 1 和条件 2.

当 A1 的值大于等于 60 时提示“及格”，采用以下代码：

```
If Range("A1").Value >= 60 Then MsgBox "及格"
```

如果条件修改为当 A1 的值大于等于 60，而且小于 80 时提示“良”，那么按照 And 的语法可以得到以下代码：

```
If Range("A1").Value >= 60 And  Range("A1").Value < 80 Then MsgBox "良"
```

VBA 中的 And 运算和工作表中的 And 函数功能一致但用法不同，不能混为一谈。通常函数爱好者初学 VBA 时，遇到与工作表函数同名的 VBA 属性、方法或者运算符时可能采用工作表函数的思路来使用 VBA 代码，这在学习初期经常带来困扰。对于这类特殊的知识点需多加留意，包括 Find、And、Or、Not、Offset、If……

2．Or 运算

Or 运算用来对两个表达式进行逻辑分析运算，当两个表达式中任意一个的运算结果为 True 时返回 True。

Or 的语法如下：

```
result = expression1 Or expression2
```

如果 A2 的值小于 60 或者 B2 的值小于 60，那么在 C2 单元格显示“补考”。按照 Or 的语法可以得到以下代码：

```
If Range("A2").Value < 60 Or Range("B2").Value < 60 Then Range("C2").Value = "补考"
```

如果 A2 的值小于 0 或者 A2 的值大于 100，那么在 B2 单元格显示“录入有误”。按照 Or 的语法可以得到以下代码：

```
If Range("A2").Value < 0 Or Range("A2").Value > 100 Then Range("B2").Value = "录入有误"
```

虽然两个条件都是针对 Range("A2").Value，但是仍然必须写两次 Range("A2").Value，引用对象时不能简化。

如果 A2、B2 或者 C2 的值小于 60，那么在 D2 单元格显示“补考”。按照 Or 的语法可以得到以下代码：

```
If Range("A2").Value < 60 Or Range("B2").Value < 60 Or Range("C2").Value < 60 Then Range("D2").Value = "
补考"
```

一个 Or 只能对两个条件执行运算，所以当有三个条件时需要使用两个 Or 运算符。第一个 Or 运算符对前两个条件执行运算，产生一个逻辑值 True 或者 False，它和第三个条件组成一对，通过第二个 Or 运算符执行逻辑运算，得到 True 或者 False。

如果还有更多的条件，每多一个条件需要加一个 Or 运算符。And 运算符同样如此。

注意：如果有多个条件，且条件不是按从左到右的顺序执行，那么可以使用括号改变执行顺序，这一点和在公式中使用括号的思路一致。

3．Not 运算

Not 运算用来对表达式进行逻辑否定运算，表示否定一个逻辑表达式。

Not 的语法如下：

```
result = Not expression
```

如果条件 expression 值为 True，那么返回值 result 则为 False；如果条件 expression 值为 False，那么返回值则为 True。

如果 A1 单元格的值是文本，那么在 A1 单元格前插入一个新单元格。按照 Not 的语法可以得到以下代码：

```
If Not IsNumeric(Range("A1")) Then Range("A1").Insert
```

其中 IsNumeric 函数用于判断参数是否为数值，当参数是数值时返回 True，否则返回 False。

本例中如果 A1 单元格的值是文本，那么 IsNumeric(Range("A1"))的返回值是 False，而 Not 运算符则可以将 False 变成 True。

所以 Not 的作用就是将 True 变成 False，将 False 变成 True。

或许读者会问，通过 IsNumeric 得到 True，再用 Not 将它转换成 False，何不只用一步直接判断是否为文本呢？答案是 VBA 只提供了 IsNumeric 来判断参数是否为数字，而没有提供 IsText 这类函数判断参数是否为文本，因此 Not 函数才有它的用武之地。

注意：在逻辑运算中除了常用的 True 和 False，也常用数字作为判断条件。在逻辑运算中，VBA 将 0 当作 False 处理，将其他数值当作 True 处理，示例如下。

```
If Range("A1").Value Then MsgBox "不等于 0" Else MsgBox "等于 0 或者空白"
```

如果 A1 单元格是空白的或者有数值 0，那么 If 将它当作 False 处理，将提示“等于 0 或者空白”；如果 A1 单元格是不等于 0 的数值，那么 If 将它当作 True 处理，将提示“不等于 0”。

6.1.4　案例解析：指定工作簿的最后开启日期

要求：工作簿只能在 2017 年 8 月 1 日前正常打开，在 2017 年 8 月 1 日之后打开工作簿时自动关闭。

根据要求分析，解决此问题需要涉及三个知识点：其一是必须让代码在开启工作簿时自动执行；其二是需要检查开启工作簿的日期是否大于 2017 年 8 月 1 日；其三是自动关闭工作簿，且需要阻止弹出是否保存对文件的更改的提示框，否则单击“取消”按钮即可阻止关闭工作簿。

根据以上分析，将以下代码写入模块中即可实现：

```
Sub auto_open()   '开启工作簿时自动执行（放置位置：模块中）
'如果当前系统日期大于等于 2017 年 8 月 1 日，那么将工作簿的 Saved 属性设置为 True
'然后关闭活动工作簿。将工作簿的 Saved 属性设置为 true 的目的是禁止 Excel 弹出提示框。
If Date >= #8/1/2017# Then ActiveWorkbook.Saved = True: ActiveWorkbook.Close
End Sub
```

如果在 2017 年 8 月 1 日前打开工作簿，那么表达式“Date >= #8/1/2017#”返回值为 False，后面的代码将不再执行；如果在 2017 年 8 月 1 日或者以后打开工作簿，那么在打开工作簿后工作簿会瞬间关闭。

代码分析

（1）代码中 Date 函数用于获取当前的系统日期，通常称之为今天的日期，但是更准确地讲是当前系统日期。如果控制面板中的日期设置不正确，那么 Date 函数无法取得今天的日期。

（2）在 VBA 中引用日期时，需要在日期前后使用井号(#)，否则表示日期的字符串“2017-8-1”将被看作减法表达式，如果采用“8/1/2017”形式来表达日期，则被看作除法表达式，唯有日期前后使用井号（#）后被当作日期处理。

（3）代码“ActiveWorkbook.Close”已经可以实现关闭活动工作簿，但是它有一个漏洞，当工作簿中有易失性函数时，不管工作簿是否修改过，关闭工作簿时都会弹出“是否保存对 xxx 工作簿的更改”的提示，如图 6-1 所示。如果在该对话框中单击“取消”按钮，那么关闭工作簿的操作将会终止。所以为了避免此问题，代码中加了“ActiveWorkbook.Saved = True”。

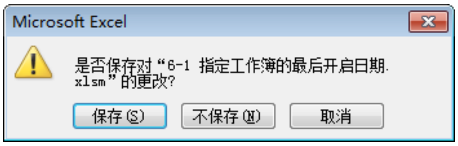


图 6–1　保存提示框

Saved 是工作簿的一个属性，表示工作簿自上一次保存后是否编辑过，将该属性赋值为 True 时表示工作簿未编辑过，Saved 属性值为 False 时表示已经编辑过。在关闭文件时，如果自上一次保存后工作簿没有编辑过就不会弹出询问提示框，所以本例将工作簿的 Saved 属性赋值为 True 后再关闭工作簿。

注意：Saved 属性赋值为 True 的目的是将工作簿标示为未编辑状态，与工作簿实际上是否编辑过无关。

（4）当 VBA 过程取名为“auto_open”且将过程代码保存在模块中时，代码将在开启该工作簿后自动执行，这是 Excel 和 Word 等软件的共同规定。

与“auto_open”对应的还有“auto_close”，表示该过程在关闭工作簿时自动执行。

本例案例文件请参考：..\ 第 6 章\ 6-1 指定工作簿的最后开启日期.xlsm

6.1.5　If...Then...Else 的块形式

If 语句包含单行模式和块形式，块形式较单行模式更灵活，并且当按条件执行多句代码时采用块形式更利于书写，也更利于他人阅读代码。

If 语句的块形式较单行模式复杂一些，语法如下。

If Condition Then ← 如果 Condition 值为 True，那么

[statements] ← Condition 值为 True 时执行的语句

[ElseIf condition-n Then ← 重新指定如果值为 True 第二个条件

[elseifstatements] ... ← 第二个条件为 True 时执行的代码

[Else ← 否则（即所有条件都不满足时）

```
[elsestatements]]   ←  所有条件都不满足时执行的代码

End If              ←  结束条件语句
```

其中 Condition 表示条件，statements 表示符合条件时需要执行的代码，可以是一句也可以是多句，还可以完全忽略。

ElseIf condition-n 表示不符合第一个条件时，为其指定新的条件，可以多层嵌套。有多少个 ElseIf 就可以指定多少个新的条件 condition-n。elseifstatements 部分表示对应的新条件的代码，符合新条件时需要执行的代码。

最后的 Else 与 elsestatements 部分表示不符合以上所有条件时需要执行的代码，是可选项。要注意每个部分都各占一行，换行与不换行时会影响代码的执行效果。

在块形式的 If 语句中，End If 部分是必选的，而 ElseIf 和 Else 部分是可选的。

所有单行模式的 If 语句都可以改为块形式的 If 语句完成，例如以下语句：

```
If Range("A2").Value < 60 Or Range("B2").Value < 60 Then Range("C2").Value = "补考"
```

如果改用块形式的 If 语句可以修改为：

```
If Range("A2").Value < 60 Or Range("B2").Value < 60 Then
  Range("C2").Value = "补考"
End If
```

由于要求中没有不满足条件时需要执行何种命令，所以 Else 和 elsestatements 部分都省略了，但是 End If 不可以省略。

如果单行模式的 If 语句如下：

```
If Range("A1").Value Then MsgBox "不等于 0" Else MsgBox "等于 0 或者空白"
```

改用块形式后效果如下：

```
If Range("a1").Value Then
    MsgBox "不等于 0"
Else
    MsgBox "等于 0 或者空白"
End If
```

块形式的 If 语句需要注意代码缩进、对齐，使其更美观，同时提升代码的可读性。

对于以下单行模式的 If 语句：

```
If Date >= #8/1/2017# Then ActiveWorkbook.Saved = True: ActiveWorkbook.Close
```

改用块形式后效果如下：

```
If Date >= #8/1/2017# Then
    ActiveWorkbook.Saved = True
    ActiveWorkbook.Close
End If
```

块形式是专为多行代码设计的。较之于单行模式中使用冒号连接多个语句,块形式有更强的可读性。

6.1.6　块形式的应用案例：创建日期批注

要求：如果活动单元格中没有批注，那么创建一个批注，在批注中写入今日日期。

随意录制一个宏就能得到创建批注的命令 AddComment， Comment.Text 属性表示批注中显示的文字，可以随意修改。在录制宏的基础上加上一句判断单元格是否存在批注的语句即可，

完整代码如下：

```
Sub 在批注中添加日期()  '放置位置：模块中
    '如果活动单元格中没有批注（ActiveCell.Comment 的类型为 Nothing 时表示不存在批注）
    If TypeName(ActiveCell.Comment) = "Nothing" Then
        '添加批注，批注中的文字是日期（由于批注不允许使用数值，所以将它转换成文本）
        ActiveCell.AddComment CStr(Date)
    Else '否则（表示有批注）
        '在活动单元格的批注后添加日期，其中 Chr(10)表示换行，日期写在下一行
        '虽然此处的 Date 同样是数值，但是它和批注中原来的字符串连后就不再是数值了
        ActiveCell.Comment.Text ActiveCell.Comment.Text & Chr(10) & Date
    End If
End Sub
```

假设图 6-2 中 C3 单元格无批注，那么选择 C3 单元格后执行过程“在批注中添加日期”，可以得到如图 6-2 所示的效果，表示创建了一个包含当前系统日期的批注。

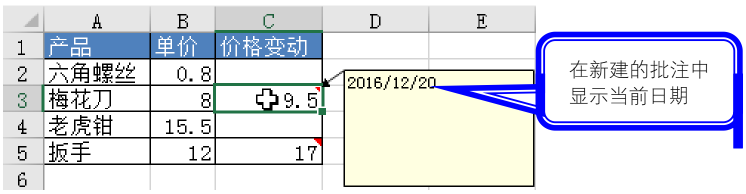


图 6–2　对没有批注的单元格创建含日期的批注

假设 C5 单元格有批注，批注内容是“原小号扳手缺货，更换中号扳手”，那么选择 C5 单元格再执行过程“在批注中添加日期”将得到如图 6-3 所示效果。

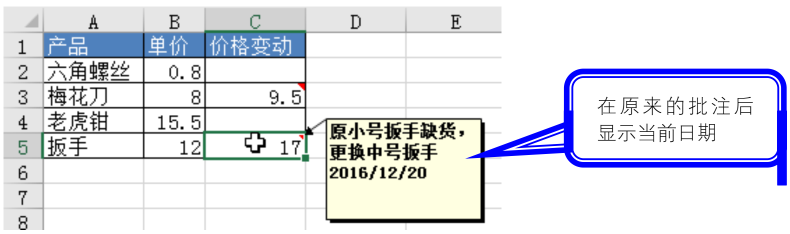


图 6–3　对有批注的单元格的批注添加当前日期

代码分析

（1）单元格是否有批注可以通过 TypeName 判断，如果有批注时返回“Comment”，没有批注时返回“Nothing”。与 TypeName 的结果执行比较的字符串需要首字母大写，其他字母小写。

（2）ActiveCell.AddComment 表示对活动单元格创建批注，AddComment 的语法如下：

```
Range.AddComment(Text)
```

参数 Text 表示批注的内容，只能是文本，不允许使用数值。

本例中 Date 表示当前的系统日期，在本质上日期值都是数值，所以用它作为批注内容时需要使用类型转换函数 CStr 将它转换成文本，否则在创建批注时会出错。

（3）Comment.Text 既是方法又是属性，如果当作方法使用，可以通过它设置批注的文本，本例中第一个 Comment.Text 就表示使用 Comment 对象的 Text 方法；如果将它当作属性，那么 Comment.Text 用于获得批注中的文字，本例中第二个 Comment.Text 表示获取批注中已经存在的文字。

（4）代码中 Chr(10)表示换行符，使用换行符可以确保日期另起一行，避免与原有的批注字

符混淆。

（5）第二次出现的 Date 函数产生的日期也是数值，但是它与原有的批注字符串合并后不再是数值，所以可以忽略转换函数 CStr。

本例案例文件请参考：..\ 第 6 章\ 6-2 创建日期批注.xlsm

6.1.7 嵌套使用 If 语句

在实际工作中经常将块形式的 If 语句嵌套使用，可以为嵌套的 If 语句指定多个条件，且在符合一个条件后再指定新的条件，也可以是不符合某条件时再指定新的条件。If 语句嵌套通常采用以下形式：

```
If Condition Then
    更多 If 语句
Else
    更多 If 语句
End If
```

此处的“更多 If 语句”既可以是块形式的：

```
If condition Then statements Else elsestatements
```

也可以是：

```
If condition Then
  statements
Else
  elsestatements
End If
```

当多个 If 语句嵌套时需要注意每层条件之间的关系。

其中的“更多 If 语句”既可以是单行模式也可以是块形式的 If 语句。

以下通过两个案例展示多个 If 嵌套的常规思路。

1．查找最小值

要求：查找工作表中的最小值，如果有多个单元格等于最小值，那么一并选中。

根据录制宏的经验可以知道，在单元格中查找值应使用 Range.Find 方法，而继续查找下一个符合条件的值则采用 Range.FindNext 方法。如果将它们配合循环语句，那么可以无限制地查找下去，直到人为地中断循环。

另外，为了让代码更具人性化，需要加一些必要的判断语句，包括判断工作表是否为空表、是否有数值。完整的代码如下。

```
Sub 查找所有最小值()                              '放置位置：模块中
   '声明一个 Range 型的对象变量，用于替代活动工作表中的已用区域
    Dim Rng As Range
    Set Rng = ActiveSheet.UsedRange               '将活动工作表的已用数据区域赋值给变量 Rng
    If IsEmpty(Rng) Then                          '判断工作表是否是空表，是空表时提示用户
        MsgBox "当前表是空表", vbOKOnly, "提示" '提示用户
    Else                                          '如果不是空表
        If WorksheetFunction.Count(Rng) Then      '如果已用区域中数值个数大于 0 个
            Dim MinValue As Long                  '声明 Long 型的变量，用于储存区域中的最小值
            MinValue = WorksheetFunction.Min(Rng) '记录 Rng 中的最小值
            '声明 3 个 Range 型变量。其中“首个对象”表示第一次查找到的单元格，
             '“下个对象”第二次及以后找到的所有单元格
            '“最终目标”表示所有找到的单元格的合集，通过 Union 将每一次查找结果合并起来
            Dim 首个对象 As Range, 下个对象 As Range, 最终目标 As Range
```

```
                '查找最小值，将找到的单元格赋予变量“首个对象”
                Set 首个对象 = Rng.Find(MinValue, , , xlWhole)
                Set 下个对象 = 首个对象          '将“下个对象”变量也赋值为第一次找到的单元格
                Set 最终目标 = 首个对象          '将“最终目标”变量也赋值为第一次找到的单元格
                Do  'Do Loop 循环语句，表示永远查找下去，因为可能符合条件者不止一个
                   '查找下一个（从第一次找到的单元格后面开始找），FindNext 的参数是参照单元格，
    '在该单元格之后继续查找
                    Set 下个对象 = Rng.FindNext(下个对象)
                    '如果找到的下一个单元格的地址和第一次找到的单元格的地址完全一样
                    If 下个对象.Address = 首个对象.Address Then
                        最终目标.Select              '那么选择变量“最终目标”所代表的区域
                        End                          '结束一切，包括结束循环、过程和释放变量的值
                    Else                             '否则（表示发现新目标）
                        '将新找到的单元格与变量“最终目标”合并
                        Set 最终目标 = Union(最终目标, 下个对象)
                    End If
                Loop
            Else                                     '否则（表示工作表中没有数值）
                MsgBox "本工作表中没有数值"          '提示用户 无数值
            End If
        End If
    End Sub
```

如果活动工作表是空白工作表，那么执行代码时将弹出如图 6-4 所示对话框；如果在只有文本的工作表中执行则会弹出如图 6-5 所示的提示框。

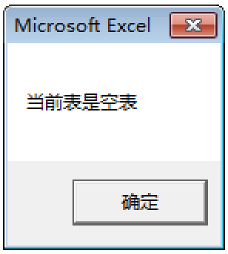
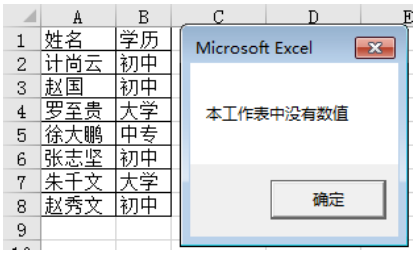


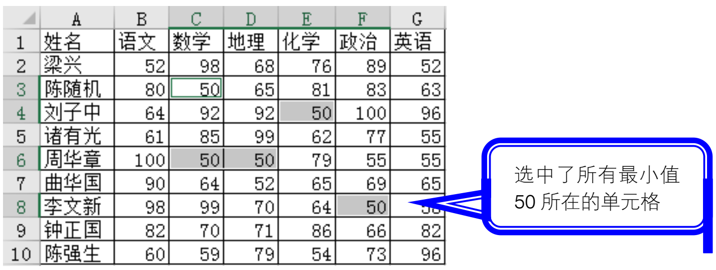
图 6–4　在空表中执行代码的结果



	A	B
1	姓名	学历
2	计尚云	初中
3	赵国	初中
4	罗至贵	大学
5	徐大鹏	中专
6	张志坚	初中
7	朱千文	大学
8	赵秀文	初中
9		

图 6–5　在只有文本的表中执行代码的结果

如果在有文本、有数值的工作表中执行，那么将瞬间选择所有最小值，如图 6-6 所示。



	A	B	C	D	E	F	G
1	姓名	语文	数学	地理	化学	政治	英语
2	梁兴	52	98	68	76	89	52
3	陈随机	80	50	65	81	83	63
4	刘子中	64	92	92	50	100	96
5	诸有光	61	85	99	62	77	55
6	周华章	100	50	50	79	55	55
7	曲华国	90	64	52	65	69	65
8	李文新	98	99	70	64	50	50
9	钟正国	82	70	71	86	66	82
10	陈强生	60	59	79	54	73	96

图 6–6　定位所有最小值单元格

代码分析

（1）对象 ActiveSheet.UsedRange 需要出现多次，所以采用变量 Rng 替代，从而简化代码，同时也对代码执行速度有所帮助。

（2） IsEmpty 函数用于判断变量是否初始化，本例中用它来判断 ActiveSheet.UsedRange。当工作表中没有任何信息时，ActiveSheet.UsedRange 属于未初始化的对象，IsEmpty 函数的返回值为 True。

（3）工作表函数 Count 用于计算区域中的数值个数，本例采用 WorksheetFunction.Count(Rng) 判断工作表中是否存在数值，当有数值时其结果一定大于 0。在逻辑运算中，大于 0 的值皆当作 True 处理。同理，代码中的 WorksheetFunction.Min 表示引用工作表函数 Min 计算区域中的最小值，如果将 Min 替换成 Max，即为计算最大值。

（4）Range.Find 方法用于在区域中查找值，支持通配符“*”和“?”，其返回值是 Range 对象，即查找目标所在的单元格。Range.Find 方法的语法请参阅本书 7.1.5 节。

本例中 Range.Find 的第一参数使用变量 MinValue，表示查找最小值，该值通过工作表函数计算而来。第四参数采用 xlWhole 表示完全匹配，如果采用 xlPart 则表示部分匹配，那么当查找 50 时会将 1500、0.50、2509 等数值也一并查找出来。

（5）Do Loop 循环表示永远循环地执行下去，如果不设定中断的条件它会耗尽所有资源，所以本例中使用了 If 语句判断“If 下个对象.Address = 首个对象.Address”，当找到的下一个目标和第一次找到的目标单元格地址相同时停止循环。Do Loop 本身可以无限制地循环，在 Do 与 Loop 中间的代码会永远执行下去，因此有必要提供中断条件。在本章第 7 节中会详细解说 Do Loop 的语法和应用技巧。

（6）Range.Find 表示第一次查找数据，而 Range.FindNext 方法则表示查找下一个，它总是配合 Range.Find 方法使用。Range.FindNext 的语法如下：

```
Range.FindNext(After)
```

其参数 After 表示在它所代表的单元格之后开始查找，而参数 After 所代表的单元格总是上一次查找到的目标单元格。例如目标上一次在 A10 单元格出现，那么 FindNext 就在 A10 单元格之后开始查找，从而避免重复，除非 Range 区域中只有一个符合条件的单元格。

（7）“Set 最终目标 = Union(最终目标, 下个对象)”表示将每一次查找到的目标单元格合并为一个对象，并赋值给变量“最终目标”。每执行一次查找，对象变量“最终目标”包含的单元格就多一个，它代表所有符合条件的区域。所以当“下个对象.Address = 首个对象.Address”返回 True 时，通过“最终目标.Select”可以选择符合条件的所有单元格。

（8）初学者可能较难理解以下三句代码：

```
Set 首个对象 = Rng.Find(MinValue, , , xlWhole)
Set 下个对象 = 首个对象
Set 最终目标 = 首个对象
```

将第一次查找到的单元格赋值给变量“首个对象”记录下来，是便于后面的比较，每找到一个新的目标时都与它的地址作比较，如果相同表示已经查找完毕，即将开始第二轮查找；将“首个对象”赋值给变量“下个对象”是为了用它作为 FindNext 的参数，因为 FindNext 执行查找时以它作为参照点，查找其后的单元格；将变量“首个对象”赋值给变量“最终目标”是为了便于合并，因为“最终目标”代表符合条件的所有单元格，所以第一次查找到目标后有必要将它赋值给变量“最终目标”。

（9）当第一轮查找结束时（即“下个对象”的地址与“首个对象”的地址一致时），不需要继续查找下去，否则会进入死循环，耗尽系统资源，所以本例采用 End 语句结束过程。

在本例中使用了 3 个 If 语句嵌套使用，在编写时要注意它们的关系，顺序不能有误。

如果需要追求效率，查找最小值通常采用数组。不过本例的思路和案例中涉及的知识点对于

任何 VBA 爱好者而言都很重要，有较多的可借鉴之处。当学会 VBA 中的数组后，读者不妨采用数组的思路改写本例代码。

本例案例文件请参考：..\ 第 6 章\ 6-3 查找最小值.xlsm

2．保存文件

在利用代码保存文件时也需要经过多次判断，本例展示与保存文件相关的 If 嵌套应用。

要求：使用代码保存活动工作簿为“汇总表.xlsm”，路径由用户自己选择。

如果仅保存文件，那么采用 SaveAs 方法即可。不过完善的代码需要考虑的问题很多，例如判断活动工作簿是否已经保存过，判断指定的路径下是否存在同名的文件，如果有同名文件则提示用户是否需要覆盖等。完整代码如下所示。

```
Sub 保存文件为汇总表()  '放置位置：模块中
    '如果活动工作簿的路径长度大于 1（表示已经保存过）
    If Len(ActiveWorkbook.Path) > 1 Then
        MsgBox "已保存过了"
    Else '否则
        '引用 FileDialog 对象，此处表示弹出对话框，允许用户选择一个文件夹
        With Application.FileDialog(msoFileDialogFolderPicker)
            .AllowMultiSelect = False       '不允许多选
            If .Show = True Then            '如果没有单击“取消”按钮（单击“取消”按钮时其值为 False）
            Dim PathName As String '声明 String 型的变量，用于存放用户指定的路径
              PathName = .SelectedItems(1)   '将选择的第一个文件夹名字赋予变量 PathName
              '如果路径右边没有“\”，就追加一个“\”
              If Right(PathName, 1) <> "\" Then PathName = PathName & "\"
              '如果已经存在同名的文件（当指定路径的文件不存在时，Dir 的返回值长度为 0）
              If Len(Dir(PathName & "汇总表.xlsm")) > 0 Then
                '提示用户已有同名文件，询问是否覆盖，如果选择“否”，那么结束过程
                If MsgBox("已存在同名文件，是否覆盖？", vbYesNo, "确认") = vbNo Then End
              End If
              '将文件另存为指定路径下的汇总表.xlsm，文件类型“Excel 启用宏的工作簿”
        ActiveWorkbook.SaveAs PathName & "汇总表.xlsm", xlOpenXMLWorkbookMacroEnabled
            End If
        End With
    End If
End Sub
```

如果活动工作簿已经保存过，那么在执行以上代码时会弹出如图 6-7 所示的提示框。

如果工作簿未保存过，则会弹出“浏览”对话框供用户选择保存文件的路径。在图 6-8 中选择了 E 盘的“生产表”文件夹，那么保存文件的完整路径就是“E:\生产表\汇总表.xlsm”。

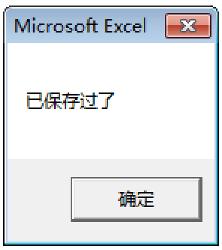


图 6-7　提示文件已保存过

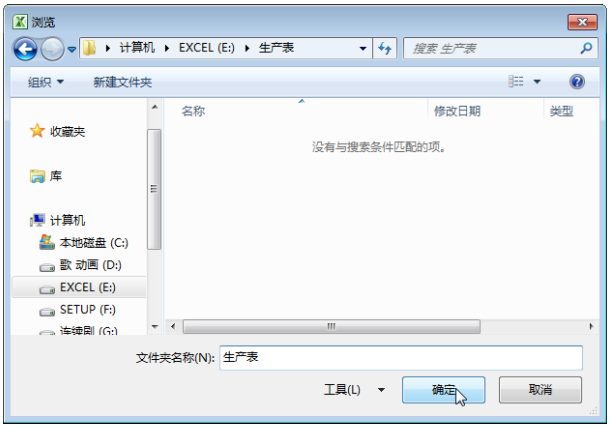


图 6–8　选择保存文件的路径

如果在 E 盘的“生产表”文件夹中已经存在“汇总表.xlsm”文件，那么会询问用户是否覆盖同名文件，如图 6-9 所示。如果单击“否”按钮那么会结束程序，不保存文件；如果单击“是”按钮则保存文件，并覆盖同名文件。

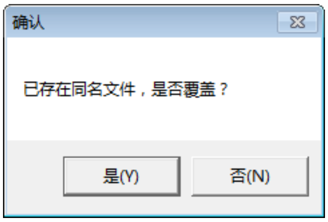


图 6–9　提示是否覆盖同名文件

代码分析

（1）Path 属性表示文件的路径，包含磁盘名称。如果文件已经保存过，那么它的路径长度必定大于 1 位。所以在本例中使用“Len(ActiveWorkbook.Path)”语句计算活动工作簿的路径长度，从而判断文件是否保存过。

（2） FileDialog 对象可以提供一个对话框，供用户选择文件或者文件夹。当对 FileDialog 使用不同参数时将获得不同功能。在表 6-1 中提供了 4 个可用的参数及其功能说明。

表 6-1　FileDialog的参数列表与功能说明

参　　数	值	功　　能
msoFileDialogFilePicker	3	调用“文件选取器”对话框。
msoFileDialogFolderPicker	4	调用“文件夹选取器”对话框。
msoFileDialogOpen	1	调用“打开”对话框。
msoFileDialogSaveAs	2	调用“另存为”对话框。

本例中采用 msoFileDialogFolderPicker 作为参数，因此 FileDialog 对象提供一个可选择文件夹的对话框，当用户选择文件夹并单击“确定”按钮后可以通过“SelectedItems(1)”获取用户所选文件夹的完整路径名称。

AllowMultiSelect 表示对话框的多选属性，赋值为 True 时表示允许按住【Ctrl】键或者【Shift】键的同时选择多个文件夹；当该属性赋值为 False 时，只能选择单个文件夹。

FileDialog.Show 方法表示显示对话框，它的返回值由用户选择的按钮决定，当在对话框中单击“确定”按钮时返回 True；当单击“取消”按钮时则返回 False。所以本例根据它的返回值判断是否单击了“确定”按钮，如果单击“取消”则结束过程。

表 6-1 中的值可以代替常数使用，例如将 msoFileDialogFolderPicker 改成 4 后，效果一致。

（3）通过 FileDialog 对象获得的路径有可能右边一个字符是“\”也可能不是“\”，当选择根目录时其路径右边有“\”，例如“C:\”。为了确保路径的右边一个字符一定是“\”，本例在条件语句中通过 Right 函数提取右边一位，判断其是否为“\”，如果没有则对其追加一个“\”。

（4）Dir 函数用于提取文件或者文件夹名称，其语法如下：

```
Dir[(pathname[, attributes])]
```

其中第一参数 pathname 表示路径，可以是文件夹路径也可以是文件路径。第二参数 attributes 代表文件属性，表示 Dir 函数只提取符合此属性的文件或者文件夹名称。

attributes 参数的可选项包括以下 7 项如表 6-2 所示，其中值和常数可以相互代替。

表 6-2　attributes参数的可选项及其他说明

常　数	值	说　明
vbNormal	0	指定不限定属性的文件（默认选项）
vbReadOnly	1	指定无属性的只读文件
vbHidden	2	指定无属性的隐藏文件
VbSystem	4	指定无属性的系统文件 在Macintosh中不可用
vbVolume	8	指定卷标文件；如果指定了其他属性，则忽略vbVolume
vbDirectory	16	指定无属性文件及其路径和文件夹
vbAlias	64	指定的文件名是别名，只在Macintosh上可用

本例中忽略了第二参数，所以相当于使用 vbNormal，表示提取文件名称，忽略其属性。

在表 6-2 中，常用的是 vbNormal 和 vbDirectory，前者表示获取文件名称，后者既可以获取文件名称，也可以获取文件夹的名称，它取决于第一参数是否包含文件名称。当第一参数包含文件名称时，取文件名称，否则取文件夹名称，示例如下。

MsgBox Dir("C:\Windows", vbDirectory)——获取文件夹名称 Windows。

MsgBox Dir("C:\Windows\explorer.exe", vbDirectory) ——获取文件名称 explorer.exe。

本书后面的章节还有更多关于 Dir 函数的应用。

在本例中“Len(Dir(PathName & "汇总表.xlsm"))”表示计算文件名称的长度，如果文件不存在，那么 Dir 函数只能提取到空文本，其长度为 0，所以在判断文件或者文件夹是否存在时采用 Len+Dir 组合即可。

（5）MsgBox 的第二参数用于指定按钮的数目、形式、图标样式、默认按钮等，信息对话框的返回值也由第二参数决定。本例中第二参数使用 vbYesNo 表示在对话框中显示“是”与“否”两个按钮，如果单击“是”按钮则其返回值为“VbYes”，否则返回“VbNo”。根据此特性可以判断用户是否在对话框中单击了“是”按钮，然后采用 If 语句根据用户的不同选择，指定不同的操作。本例中的思路是：如果单击“否”按钮则结束过程。

（6）Workbook.SaveAs 方法表示另存工作簿，其语法如下：

```
WorkBook.SaveAs(FileName, FileFormat, Password, WriteResPassword, ReadOnlyRecommended, CreateBackup, AccessMode, ConflictResolution, AddToMru, TextCodepage, TextVisualLayout, Local)
```

参数 FileName 表示文件名称，需要指定路径。如果忽略路径，则表示保存在 Excel 选项对话框中指定的默认文件位置。图 6-10 即为“Excel 选项”对话框中的用于设置默认文件位置的选

项，笔者近期操作最多的文件夹是“E:\书\第十本\案列文件”，所以将它设置为默认文件位置。

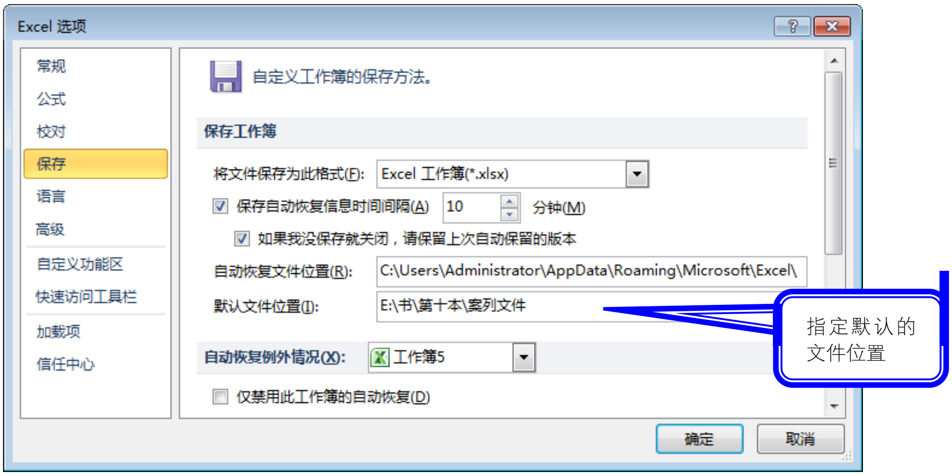


图 6–10　默认文件位置

第二参数 FileFormat 表示文件格式。Excel 2016 支持的文件格式较多，可以在 VBA 的在线帮助中搜索“XlFileFormat 枚举”关键字，从而获得所有文件格式的说明。

Excel 2016 中常用的三种文件格式及其说明如表 6-3 所示。

表 6-3　Excel 2016 常用格式一览

文件格式	后缀名	说　明
xlOpenXMLWorkbook	.xlsx	常规格式
xlOpenXMLWorkbookMacroEnabled	.xlsm	启用宏的工作簿
xlAddIn	.xlam	加载宏

本例中将文本保存为 xlsm 格式，因此格式名称采用 xlOpenXMLWorkbookMacroEnabled。

第三参数 Password 表示保存文件时对文件设置的打开密码。其余参数不常用，有兴趣的读者可以通过查询帮助了解详情。

本例案例文件请参考：..\ 第 6 章\ 6-4 保存文件.xlsm

6.1.8　If 语句的常见错误与防错之法

在使用 If 语句的过程中初学者常犯三个错误，本节一一说明，并提供防错之法。

1．忘记 End If

当使用块形式的 If 语句时，如果 If Then 与 End If 之间只有几行代码，通常不会忘记写 End If，如果其间有十多行或者几十行代码那么忘记写 End If 的可能性就大得多了。在代码中缺少 End If 时会产生如图 6-11 所示的提示框，解决办法是在代码中适当的地方添加 End If。

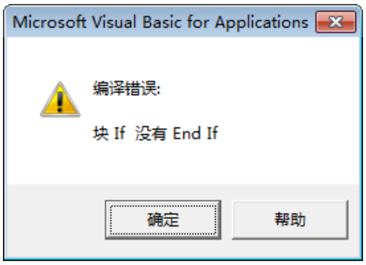


图 6–11　错误提示

事实上后期的补救总显得效率低下，前期书写完善才是上策。笔者的建议是在书写代码时写完 Then 后马上另起一行录入 End If，然后再返回前一行继续录入其他代码，此方式可以百分百杜绝忘记书写 End If 的问题。例如在以下代码中，书写顺序为第一行、第三行、第二行：

```
If A > B Then
  MsgBox C
End If
```

2．statements 参数放错了地方

在块形式的 If 语句中，statements 参数在第二行，如果放在第一行则会出现如图 6-12 所示的提示框。

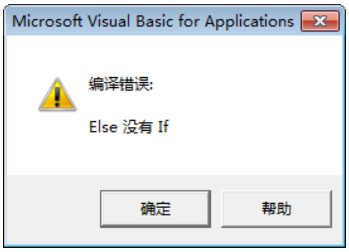


图 6–12　错误提示

正确的的块形式 If 语句应按如下方式书写：

```
If ActiveSheet.Name = "Sheet3" Then
    Range("A1") = "VBA"
    Range("B1") = "好犀利"
Else
    Range("A1") = "支持楼上"
End If
```

而当代码写为以下形式时将会出错：

```
If ActiveSheet.Name = "Sheet3" Then Range("a1") = "VBA"
    Range("B1") = "好犀利"
Else
    Range("A1") = "支持楼上"
End If
```

在书写块形式的 If 语句时，Then 后面必须换行。

有时会因为语句过于简单而将块形式的 If 语句改成单行模式，从而将第二行的代码移到与 Then 同行的位置，此时需要记住的是必须删除 End If，否则必定出错。

3．未正确使用冒号

使用冒号可以将多行代码放在一行中执行，但是在某些情况下，将多行代码放到一行中会产

生与预期截然不同的效果。例如执行以下过程将不产生任何反应，因为第一个 If 语句的条件不成立，然后执行第二行“Exit Sub”，它会终止程序，从而使后面的两句代码都不执行：

```
Sub test()    '放置位置：模块中
    If Date < #10/1/2017# Then MsgBox "未到国庆节"
    Exit Sub
    If Date = #10/1/2017# Then MsgBox "今天是国庆节"
    If Date > #10/1/2017# Then MsgBox "国庆节过了"
End Sub
```

如果将代码“Exit Sub”合并到第一句中去，则会执行后面两句代码，从而提示“国庆节过了”。

```
Sub test()    '放置位置：模块中
    If Date < #10/1/2017# Then MsgBox "未到国庆节":   Exit Sub
    If Date = #10/1/2017# Then MsgBox "今天是国庆节"
    If Date > #10/1/2017# Then MsgBox "国庆节过了"
End Sub
```

所以在使用 If 条件语句时需要分清楚每条语句在什么条件下可以执行，什么条件下不可以执行。冒号可以将多行语句合并到一行中，但是并非所有情况下都可以合并，在合并前要仔细辨别被合并的多行语句是否存在并列关系。

6.2 Select Case 语句解析

Select Case 语句也是条件语句，专为多条件判断而存在。它较 If Then 更复杂，但在应对多条件判断时却更得心应手。

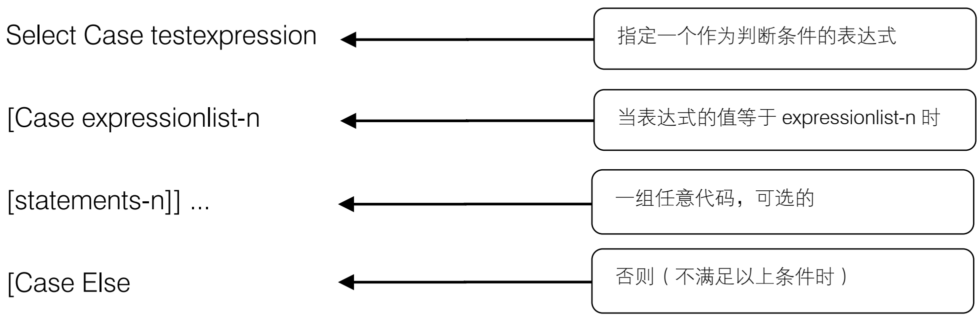
6.2.1 Select Case 语句的价值

Select Case 语句和 If 语句一样都用于按不同条件执行不同的语句，但 Select Case 语句更偏重于多条件应用，它在进行多条件判断时写法简洁许多。其次 Select Case 语句在以数值范围作为条件时也比较得心应手，通过简单的代码就能指定多个条件范围。

当然，并非 Select Case 语句有如此多优势就能取代 If 语句，Select Case 与 If 各有自己的应用领域，可以互补长短。

6.2.2 Select Case 基本语法

Select Case 语句的语法如下。



[elsestatements]] ← 另一组代码，可选的

End Select ← 结束条件语句

其中 testexpression 部分是一个表达式，其计算结果通常为逻辑值或者数值。

expressionlist-n 部分表示指定条件范围，当符合此范围时执行其后一行对应的语句。expressionlist-n 有三种形式：expression、expression To expression 和 Is comparisonoperator。

◆ Expression

它表示直接指定一个值，例如“Case　5”表示等于 5。

◆ expression To expression

它用于指定起始值和终止值，例如“Case 5 To 10”表示从 5 到 10 的数值范围。

◆ Is comparisonoperator

Is 通常配合比较运算符来指定条件范围，例如“Case　Is <5”表示小于 5 的数值范围。

statements-n 部分表示符合条件时需要执行的语句。条件和对应的语句都可以有多个。

elsestatements 部分表示不符合前面的所有条件时需要执行的语句。Case Else 和 elsestatements 是可选项，但是在编写代码时尽量不要忽略这两个项，以确保程序能应对一些不可预见的状况。

由于 Select Case 可以执行多重条件，当表达式的值符合多个条件时，Select Case 语句只执行第一次设定的条件所对应的代码。

Select Case 功能强大，但使用方法比 If 语句更复杂。现提供一个简单的案例应用，加深读者对 Select Case 的理解。

假设根据当前时间判断现在是凌晨、上午、正午、下午还是晚上。当时间为 1 到 7 点钟时表示凌晨，8 到 11 点表示上午，12 点表示正午，13 到 20 点表示下午，21 到 24 点表示晚上。使用以下代码可以满足要求：

```
Sub 时间()  '放置位置：模块中
    Select Case Hour(Now)
    Case 1 To 7
        MsgBox "凌晨"
    Case 8 To 11
        MsgBox "上午"
    Case Is = 12
        MsgBox "正午"
    Case 13 To 20
        MsgBox "下午"
    Case Else
        MsgBox "晚上"
    End Select
End Sub
```

在 VBA 中 Now 表示当前日期和时间，Hour 函数可以从中单独提取小时数出来。所以代码“Select Case Hour(Now)”表示以当前时间的小时数作为判断条件。

“Case 1 To 7”表示小时数在 1 到 7 之间，包含 1 和 7。假设满足此条件则提示“凌晨”。

“Case 12”表示等于 12，所以也可以写作“Case Is = 12”。

“Case Else”表示不符合以上所有条件时的例外情况，在本例中表示 21 点到 24 点。

事实上，本例还有另一种写法：

```
Sub 时间2()  '放置位置：模块中
    Select Case Hour(Now)
    Case Is > 20
        MsgBox "晚上"
    Case Is > =13
        MsgBox "下午"
    Case Is = 12
        MsgBox "正午"
    Case Is > 7
        MsgBox "上午"
    Case Else
        MsgBox "凌晨"
    End Select
End Sub
```

此过程采用了全新的思路，抛弃“To”，改用比较运算符来限制范围。新思路的优点在于代码更简单，缺点在于条件的出现顺序不能打乱，否则将无法得到正确的结果。Select Case 检查条件时总是从上向下的，如果将“>=13”和“>20”交换位置，那么 22 点也将被识别为下午，而不再是晚上了。而当采用“To”罗列条件时，不限顺序，任何一个条件放在前面都不影响结果。

此外，在条件中使用大于号“>”需要从后向前罗列条件，与题目中的条件顺序相反，不太符合习惯。所以这里再提供一个使用小于号“<”罗列条件的思路，代码如下：

```
Sub 时间3()  '放置位置：模块中
    Select Case Hour(Now)
    Case Is < 8
        MsgBox "凌晨"
    Case Is < 12
        MsgBox "上午"
    Case Is = 12
        MsgBox "正午"
    Case Is < 21
        MsgBox "下午"
    Case Else
        MsgBox "晚上"
    End Select
End Sub
```

在此思路中，假设当前时间是 9 点，它既符合第二个条件“<12”，又符合第 4 个条件“<21”，但是 Select Case 语句的规则是从上向下执行，只要遇到一个符合的条件就结束比较，所以其结果总是正确的，不会出现混乱。

在使用比较运算符时重点在于罗列条件时需要提供正确的顺序。

读者可以观察三种思路的差异，了解 Select Case 语句关于数值范围的多种表达方式。

本例案例文件请参考：..\ 第 6 章\ 6-5 Select Case 的基本应用思路.xlsm

6.2.3 多条件应用案例

Select Case 专为多条件判断而存在，而实际工作中也确实存在较多多条件判断的应用情景，因此 Select Case 语句的应用较为频繁。下面展示三个多条件判断的案例。

1．判断成绩分布状况

要求：假设根据 A1 的成绩返回评语。当成绩大于 0 分且小于 60 分时，在 B1 单元格返回“不及格”；　当成绩为 60 分及以上并且在 100 分以下时，在 B1 单元格显示“及格”；当成绩为 100 分时，在 B1 单元格显示“满分”，其他情况则显示“成绩有误”。根据此要求可以得到以下代码：

```
Sub 判断成绩范围()                          '放置位置：模块中
    Select Case Range("A1").Value           '以 A1 的值作为判断对象
    Case Is < 0, Is > 100                   '假设小于 0，或者大于 100
        Range("B1").Value = "成绩有误"       '返回“成绩有误”
    Case 100                                '假设等于 100
        Range("B1").Value = "满分"           '返回“满分”
    Case Is >= 60                           '假设“大于等于 60”
        Range("B1").Value = "及格"           '返回“及格”
    Case Else                               '其他情况
        Range("B1").Value = "不及格"         '返回“不及格”
    End Select                              '结束条件语句
End Sub
```

代码分析

（1）Case Else 语句不可以使用 And 和 Or 运算符，只能使用逗号实现 Or 运算符的同等功能，但没有其他办法实现 And 运算符的功能。

鉴于此种情况，题目要求是大于 0 而且小于 60，但是编写代码时需要改用新的思路。本例中首先处理小于 0 和大于 100 的范围，然后依次为 100、大于等于 60 及其他情况。

也就是说，由于 Case Else 语句不支持 And 运算符，经常需要改变条件的顺序，否则可能无法达成需求。所以本例的要求是小于 0 分和大于 100 分的为例外项，而通过 Case Else 语句实现功能时却将大于等于 0、小于 60 的范围作为例外项。

（2）“Is < 0, Is > 100”表示小于 0 分或者大于 100 分，使用逗号相当于实现 Or 运算符的功能，但是在 Case Else 语句中不允许直接使用 Or 运算符。

（3）“Case 100”必须放在“Case Is >= 60”前面，因为 Case Else 语句总是从上而下执行的，遇到第一次满足的条件后停止。

（4）由于成绩不可能有两位小数，所以“大于 0 分且小于 60 分”这个条件也可以使用“Case 0 To 59.9”来体现，那么整个过程代码的排列方式又大大不同了，这个问题交给读者完成吧！

本例案例文件请参考：..\ 第 6 章\ 6-6 Select Case 案例应用.xlsm

2．在区域中录入一周的英文单词

要求：从活动单元格开始录入代表一周的 7 个英文单词，允许选择简写还是完整书写方式。

首先分析需求，题目中要求提供两个选项，那么可以使用 Msgbox 实现，其第二参数使用 vbYesNoCancel 或者 vbYesNo。也可以使用 Application.InputBox 产生一个输入框，让用户在其中输入参数，所以本例提供两种思路。

思路一的代码如下：

```
Sub 产生表示一周七天的英文单词()              '放置位置：模块中
'引用活动单元格开始的 7 个单元格
    With ActiveCell.Resize(7, 1)
        '以 Msgbox 的返回值作为条件
        Select Case MsgBox("简写请选“是”" & Chr(13) & "完整书写请按“否”", vbYesNoCancel, "确认书
```

```
写方式")
            Case vbYes                                           '如果选择了是
                .FormulaArray = "=text(row(2:8),""DDD"")"        '录入数组公式
                .Value = .Value                                  '将公式转换成值
            Case vbNo                                            '如果选择了“否”
                ActiveCell.Resize(7, 1) = "=text(row(2:8),""DDDD"")"  '录入数组公式
                .Value = .Value                                  '将公式转换成值
            End Select
        End With
    End Sub
```

在执行以上代码后将会弹出如图 6-13 所示对话框，允许单击“是”或“否”按钮来决定单词采用简写还是完整书写。假设单击“否”按钮，将产生如图 6-14 所示结果。

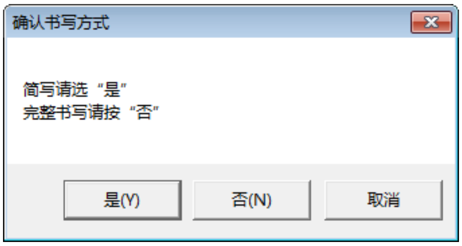


图 6–13　确定录入单词的方式

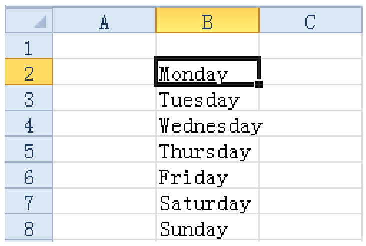


图 6–14　录入结果

代码分析

（1）代码“MsgBox("简写请选“是”" & Chr(13) & "完整书写请按“否”", vbYesNoCancel, "确认书写方式")”能产生一个信息框，展示文字信息，同时它也有返回值，包括是、否和取消三个按钮，分别对应常量 vbYes、vbNo 和 vbCancel。

（2） MsgBox 总有返回值，而且不可能产生预设值以外的值，因此在 Select Case 语句中可以忽略 Case Else 部分，它不存在例外情况。

（3）FormulaArray 表示数组公式，它是 Range 对象的一个属性，用于在单元格或者区域中产生数组公式。“.FormulaArray = "=text(row(2:8),""DDD"")"”表示在活动单元格开始的 7 个连续单元格中输入数组公式“=text(row(2:8),"DDD")”。

公式“=text(row(2:8),"DDD")”中间有两个双引号，而要将此公式赋值给 Range 的 FormulaArray 属性时，需要在公式首尾再添加引号。

VBA 中规定需要产生文本的引号时需要使用两个双引号，即代码中两个“""”可以产生一个“"”，因此在单元格中产生单个双引号要用以下代码：

```
    ActiveCell = """"
```

其中首尾的双引号是真正的引号，具有引号的作用，表示它里面是文本。而里面的两个双引号是文本，用于产生一个双引号。所以本例中 TEXT 函数的第二参数本身是“"DDD"”，即首尾只各需要一个双引号，但在书写代码时需要使用四个双引号。

（4）公式“=text(row(2:8),"DDD")”的功能是产生星期一到星期日的简写英文单词，其中“row(2:8)”表示 1900 年 1 月 2 日到 1900 年 1 月 8 日，刚好是周一到周日，所以 Text 函数将它们转换成英文后即可满足题目的要求。

（5）“.Value = .Value”表示将区域中的公式转换成值，取消此句代码则能在区域中看到数组公式。

本例案例文件请参考：..\ 第 6 章\ 6-6 Select Case 案例应用.xlsm

思路二的代码如下：

```
Sub 产生表示一周七天的英文单词 2()  '放置位置：模块中
'引用活动单元格开始的 7 个单元格
    With ActiveCell.Resize(7, 1)
ReStar: '设置一个标签
        '以 Application.InputBox 对话框的输入值作为条件
        Select Case Application.InputBox("输入 1：简写形式" & Chr(13) & "输入 2：完整书写", "确认书写方式", , , , , , 1)
        Case 1                                              '如果输入 1
            .FormulaArray = "=text(row(2:8),""DDD"")"       '录入数组公式
            .Value = .Value                                 '将公式转换成值
        Case 2                                              '如果输入 2
            ActiveCell.Resize(7, 1) = "=text(row(2:8),""DDDD"")"  '录入数组公式
            .Value = .Value                                 '将公式转换成值
        Case Else                                           '否则
            MsgBox "只能输入 1 或者 2": GoTo ReStar         '提示，并回到标签处继续执行
        End Select
    End With
End Sub
```

执行以上代码后会弹出图 6-15 所示的对话框，允许用户输入数值。假设输入 1 将产生简写单词，输入 2 则产生完整书写的单词，输入其他数值则会产生如图 6-16 所示的提示框。



图 6-15 确定录入单词的方式



图 6-16 错误时的提示

代码分析

（1）Application.InputBox 可以产生一个具有校验功能的输入框，允许用户在框中输入指定的字符。本例中最后一个校验参数使用 1 表示只允许输入数值，如果输入了文本则会弹出“无效的数字”提示框。

（2）由于 Application.InputBox 可以输入 1 和 2 以外的数字，因此有必要通过 Case Else 语句处理例外情况。

（3）Application.InputBox 方法和 InputBox 函数都能产生输入框，但是工作中尽量使用 Application.InputBox 方法，因为它具有校验功能，可以强制用户只能输入指定类型的字符。

6.3 IIf 函数

条件语句 If Then 和 Select Case 语句可以应付所有多条件判断的应用， 但是它们都不是函数，不能直接返回值。IIf 函数可以弥补这个缺陷，它可以根据条件返回指定的值。

6.3.1　IIf 函数语法解析

IIf 函数和工作表函数 If 的功能与用法都一致，可以根据表达式的计算结果来决定返回两个值中的一个。IIf 函数的语法如下：

```
IIf(expr, truepart, falsepart)
```

第一参数 expr 表示用来判断真伪的表达式，返回值通常是 True 或者 False，如果表达式的返回值是文本则会执行失败，提示“类型不匹配”；如果表达式的返回值是 0 则当作 False 处理，那么 IIf 函数返回第三参数的值；如果表达式的返回值是不为 0 的数值，那么当作 True 处理，此时函数的返回第二参数的值。

IIf 的三个参数都是必选参数，不能忽略任何参数，这一点和工作表函数 If 不同。

如果 A1 的值大于 800 则返回“产量合格”，否则返回“产量不足”，使用 IIf 函数实现可以使用以下代码：

```
Sub IIf 应用()              '放置位置：模块中
  MsgBox IIf(Range("A1").Value > 800, "产量合格", "产量不足")
End Sub
```

当需要返回值时，IIf 较 If Then 或者 Select Case 语句更方便。例如 If Then 语句代替 IIf 实现以上需求，需要用如下方式编写代码，这么做显然不够简洁。

```
Sub If_Then 代替 IIf()  '放置位置：模块中
  If Range("A1").Value > 800 Then MsgBox "产量合格" Else MsgBox "产量不足"
End Sub
```

6.3.2　IIf 函数案例应用：判断 Excel 的版本号

IIf 函数仅用于计算，对于满足条件时就插入行或者改变单元格的背景色等操作，就超出了它的职能范围。以下案例可以帮助读者理解 IIf 的使用技巧，案例中需要函数 IIf 嵌套使用。

要求：计算当前使用的 Excel 的版本是 Excel 2003、Excel 2007、Excel 2010、Excel 2013 还是 Excel 2016。

根据题目要求以及 IIf 函数的语法，编写代码如下：

```
Sub Excel 版本号()              '放置位置：模块中
    Dim Ver As String           '声明一个 String 型的变量，用于表示版本号
   '获取 Excel 的版本号，并将它赋值给变量。Excel 的开发代号采用 11.0、12.0 这种编号方式，
'所以再用 IIf 函数转换成 Excel 2003 这种形式的版本号
    Ver = Application.Version
    '通过 IIf 函数转换开发代号为符合大众阅读习惯的版本号
    MsgBox IIf(Ver = "11.0", "Excel 2003", IIf(Ver = "12.0", "Excel 2007", IIf(Ver = "14.0", "Excel 2010", IIf(Ver = "15.0", "Excel 2013", IIf(Ver = "16.0", "Excel 2016", "您的 Office 版本比较古老")))))
End Sub
```

代码分析

（1） Application.Version 表示 Excel 的开发代码，Excel 2003 对应“11.0”，Excel 2007 对应“12.0”，Excel 2010 对应“14.0”，Excel 2013 对应“15.0”，Excel 2016 对应“16.0”。西方人对 13 这个数字有所忌讳，所以避开了“13.0”这个代号。

（2）Application.Version 需要在代码中使用 5 次，所以有必要将它赋值给变量，然后调用变量 5 次，从而提升代码的执行速度。

（3）在 IIf 函数嵌套使用时，通常对第三参数嵌套，但第二参数也允许使用 IIf 函数嵌套。

（4）IIf 函数的第三参数非可选参数，在任何情况下都不能忽略。

提示：IIf 函数的条件也可以使用 And 或者 Or 运算符。

本例案例文件请参考：..\ 第 6 章\ 6-7 IIF 函数案例应用.xlsm

6.3.3　IIf 函数的优缺点

IIf 函数可以根据预设的条件返回值，可以多层嵌套使用，这样使用起来比较顺手。

不过 IIf 函数的缺点也较明显，主要体现在以下三方面。

1. 没有可选参数

工作表函数 If 和 VBA 中的 If Then 语句都有可选参数，所以 VBA 开发者在编程过程中容易按惯例操作，错误地省略了 IIf 的第三参数，那么执行代码时将产生如图 6-17 所示的错误提示。

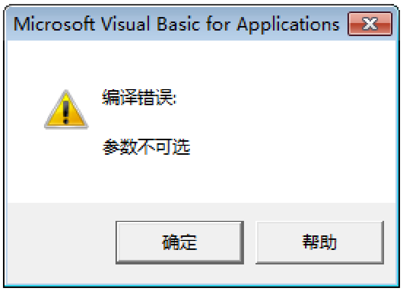


图 6–17　错误提示

"参数不可选"表示忽略了某个必选参数，将参数补充完整即可解决此问题。

2. 总是检查第二参数和第三参数的值

IIf 的第二参数和第三参数都是必选参数，而且不管第一参数的值是 True 还是 False，都会对第二参数和第三参数执行计算，所以第二参数或者第三参数有错误时一定会出错。

例如不论 A1 的值是否等于 0，以下语句总是执行出错。

```
MsgBox IIf(Range("a1").Value > 0, Range("b1") / Range("a1"), Range("b1") / 100)
```

虽然代码中限制了 A1 单元格的值大于 0 时才以 A1 的值作为除数，否则以 100 作为除数，但事实上不管 A1 的值是多少，VBA 都会分别计算 IIf 的第二、第三参数，从而导致运行出错。如果采用工作表函数 If 或者 If Then 语句则可以避免此问题。

3. 计算结果是错误值时无法执行代码

工作表函数可以返回错误值，但是 VBA 中的 IIf 函数在遇到错误时不会返回错误值，而是直接弹出错误提示，同时中断程序。

6.4　For Next 语句解析

For Next 是工作中应用最频繁的一种循环语句，它表示以指定次数重复执行一组语句，从而扩展代码的功能，使原来的单一操作得以批量执行。本节展示 For Next 语句的作用、语法和案例。

6.4.1　循环语句的作用

变量对于 VBA 而言极其重要，其中最能体现变量的优势的地方是配合循环语句 For Next 使用。

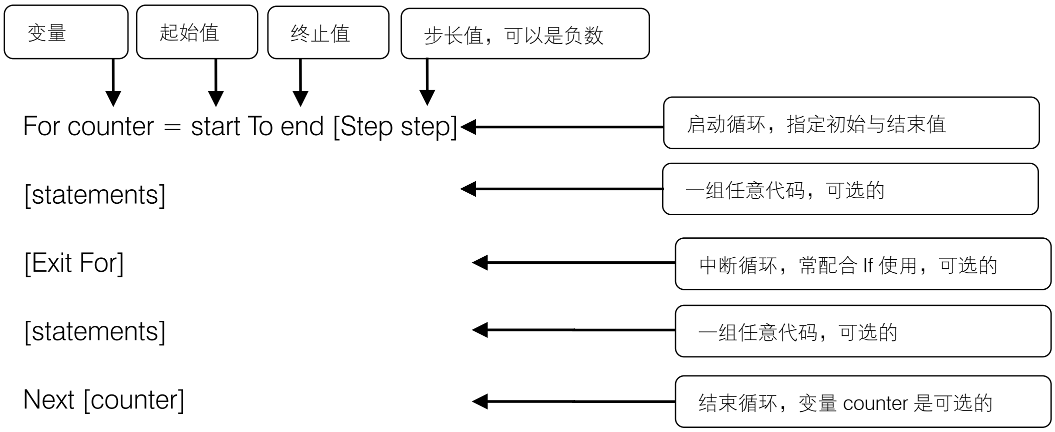
在 VBA 中使用循环语句可使代码反复执行，直到完成所有需求后自动或者手动终止，这对于制表工作而言有着较变量本身更重要的意义。

例如对插入行的代码添加循环语句后可以实现批量插入多行，或者隔一行插入一行，或者在每个工作表的相同地方都插入行……

应用循环语句后，代码的应用前景将完全呈现出一片新天地。

6.4.2　For Next 语句基本语法

For Next 语句表示以指定次数重复执行一组语句，它的语法如下。



上述语法中各部分的含义如表 6-4 所示。

表 6-4　For Next语法含义分析

部　　分	必选/可选	描　　述
counter	必选	用做循环计数器的数值变量。这个变量不能是布尔值Boolean或数组元素
start	必选	counter的初始值，或者称之为最小值
end	必选	counter的终止值，或者称之为最大值
step	可选	counter的步长值，表示变量单次累加的数量。如果没有指定此参数则其默认为1
statements	可选	放在For和Next之间的一条或多条语句，它们将被执行指定的次数

其中以下几点值得注意。

其一，计数器 counter 表示一个递增或者递减的变量，每循环一次变量的值就变化一次，直到循环结束，或者人为中断过程。

其二，步长值 step 表示变量单次递增或者递减的数量，默认值为 1。在初始值与终止值一定的前提下，步长值越小，循环语句执行的次数越多。

其三，如果终止值小于初始值，步长值必须使用负数；如果终止值大于初始值，步长值必须使用正数；如果步长值使用 0 则会使循环语句进入死循环，永远不会结束，从而耗尽系统资源。

其四，循环语句的循环次数等于终止值减去起始值，然后除以步长值再加 1。以下为示例。

For i = 2 To 5——循环次数等于（5–2）/1+1，即 4 次。

For i = 2 To 5 Step 0.3——循环次数等于（5–2）/0.3+1，即 11 次。

当终止值小于起始值时，如果忽略了步长值，那么循环语句不会执行。

其五，当执行完所有循环语句后，步长值 step 会累加到计数器 counter 中。也就是说计数器

在循环结束后的值不是循环语句的终止值，而是在该基础上加上步长值 step。例如：执行以下代码后，变量 *i* 的值是 22，不是 20。

```
Sub test()                         '放置位置：模块中
  For i = 10 To 20 Step 2          '从 10 到 20，步长值为 2
    Next i                         '进入下一轮循环，直到变量 i 的值大于 20 时停止
    MsgBox i                       '提示变量 i 的值
End Sub
```

其六，步长值会影响代表计数器的变量。例如以下代码：

```
Sub test()                         '放置位置：模块中
  Dim i As Byte                    '声明一个 Byte 型的变量
    For i = 200 To 100 Step -1     '从 200 到 100 倒置循环，步长值为–1
  Next i                           '进入下一轮循环，直到变量 i 的值小于 100 时停止
End Sub
```

此过程中虽然变量 *i* 的值在 200 到 99 中变化，但是步长值使用了负数，它超出了 Byte 数据类型的有效范围，所以执行代码时会弹出“溢出”的错误提示。

6.4.3　步长值对循环结果的影响

循环语句中的步长值直接影响循环语句的执行次数。

以下新建工作表的语句“Sheets.Add(after:=Sheets(Sheets.Count)).Name = i”可执行 10 次，分别创建名为 1 到 10 的工作表：

```
Sub 创建工作表()                    '放置位置：模块中
For i = 1 To 10                    '从 1 到 10 循环，步长值为 1
        '新建工作表，放在最后的位置，其命名由变量 i 决定
        Sheets.Add(after:=Sheets(Sheets.Count)).Name = i
    Next i                         '进入下一轮循环，直到变量 i 的值大于 10 时停止
    MsgBox i                       '提示变量 i 的值
End Sub
```

以下新建工作表的语句“Sheets.Add(after:=Sheets(Sheets.Count)).Name = i”可执行 5 次，分别创建名为 11、13、15、17、19 的工作表：

```
Sub 创建工作表()                    '放置位置：模块中
    For i = 11 To 20 Step 2        '从 11 到 20 循环 5 次，步长值为 2
  '新建工作表，放在最后的位置，其命名由变量 i 决定
        Sheets.Add(after:=Sheets(Sheets.Count)).Name = i
    Next i                         '进入下一轮循环，直到变量 i 的值大于 20 时停止
    MsgBox i                       '提示变量 i 的值
End Sub
```

上面两个案例足以说明步长值对循环语句的影响。

6.4.4　For Next 循环语句应用案例

For Next 循环语句在工作中应用面极广，不过思路总是一致的，不会有大的差异。本小节通过 3 个案例展示 For Next 循环语句的常规思路。

1．累加 1 到 100 之间的奇数

要求：将 1 到 100 之间的所有奇数逐一累加，获取其合计值。

奇数是指整除 2 以后有余数的数值，例如计算 1 到 100 之间的奇数之和有两种思路，其一是逐一检查 1 到 100 的每个数值是否属于奇数，如果是奇数则参与求和，否则略过；其二是将循环

语句的步长值设置为 2，自动跳过偶数值。思路一的代码如下：

```
Sub 汇总 1 到 100 的奇数()                          '放置位置：模块中
    Dim Item As Byte, Total As Integer              '定义两个变量，一个用于计数器，一个保存合计值
    For Item = 1 To 100                             '从 1 到 100
        If Item Mod  2 Then Total = Total + Item    '如果计数器的值是奇数则累加到变量 Total 中
    Next Item                                       '执行下一轮循环
    MsgBox Total                                    '报告最终结果
End Sub
```

执行以上代码可得到结果 2500。

代码分析

（1）计数器 Item 在 1 到 101 之间变化，所以数据类型采用 Byte，而变量 Total 的最终计算结果超过了 Byte 的有效范围，所以改用 Integer。

（2）本例中忽略了步长值 Step，表示采用默认值 1。

（3）VBA 中判断一个数值是否是奇数可借用 MOD 运算符实现。要注意这里的 Mod 是运算符，而不是函数，和工作表函数 Mod 是不同的。“Item Mod 2”表示计算变量 Item 除以 2 的余数。

（4）If 的第一参数是数值时，零值当作 False 处理，非零值当作 True 处理，所以在本例中直接采用“Item Mod 2”作为参数，而不需要通过“Item Mod 2 = 1”转换成 True。

（5）在循环之前，变量 Total 的值是 0，每循环一次，代码“Total = Total + Item”能使变量 Total 在原值基础上累加 Item 的值，所以变量 Total 的值其实是 1、4、9、16……逐渐累加。

思路二代码如下：

```
Sub 汇总 1 到 100 的奇数 2()            '放置位置：模块中
    Dim Item As Byte, Total As Integer  '定义两个变量，一个用于计数器，一个保存合计值
    For Item = 1 To 100 Step 2          '从 1 到 100,步长值为 2
      Total = Total + Item              '如果计数器的值是奇数则累加到变量 Total 中
    Next Item                           '执行下一轮循环
    MsgBox Total                        '报告最终结果
End Sub
```

执行此代码能获得与上一个过程相同的结果。

代码分析

（1）本例中计数器的初始值是 1，步长值为 2，那么它将在 1、3、5、7、9……99 之间变化，所以直接对计数器求和即可得到最终结果。

（2）本例中步长值为 2，那么代码“Total = Total + Item”执行的次数是 50 次，所以在效率上，思路二较高。

事实上，计算 1 到 100 的奇数之和还有更高效的办法，仅用一句即可完成，不过本例重点在于展示循环语句，读者通过案例能了解 For Next 的应用思路，以及了解步长值对代码执行效率的影响。

本例案例文件请参考：..\ 第 6 章\ 6-8 For Next 应用.xlsm

2．删除空白单元格所在行

要求：以 B 列为基准，将所有空单元格所在行全部删除。

检查单元格是否空白，采用 Len 函数计算其长度即可，而删除单元格可用 Range.Delete 方

法，两者配合循环语句可以实现批量删除空白单元格所在行。完整代码如下：

```
Sub 删除空白单元格所在行()                    '放置位置：模块中
    '定义两个变量，前者表示循环语句的计数器，后者用于存放所有已删除行的行号
    Dim Item As Integer, RowName As String
    '从 B 列最后一个非空单元格的行号开始，到第一行结束，步长值为-1
    For Item = Cells(Rows.Count, "B").End(xlUp).Row To 1 Step -1
        If Len(Cells(Item, "B")) = 0 Then      '如果单元格中没有字符
            Rows(Item).Delete                  '那么删除整行
            RowName = RowName & Chr(13) & Item '记录被删除的行
        End If
Next Item
'如果变量 RowName 非空，则报告所有已删除的行的行号
If Len(RowName) > 0 Then MsgBox "以下行已成功删除：" & RowName
End Sub
```

执行代码时，如果 B 列存在空白单元格，那么程序会删除所有空白单元格所在行，并提示被删除行的行号；如果不存在空白单元格则不会有任何反应。

代码分析

（1）删除行这种操作不宜采用从前向后、逐个删除的思路编写循环代码，因为删除行后会破坏原来的布局，从而造成遗漏。例如第 2 行和第 3 行都符合条件，当删除第 2 行后原来的第 3 行自动上移，变成了第 2 行，此时再检查第 3 行其实是在检查原来的第 4 行。所以本例采用从后向前循环的方式，让 For Next 的终止值小于初始值，且步长值为–1，从而解决了因破坏数据源的布局而导致的操作失误问题。

（2）“Rows(Item).Delete”表示删除行，行号由变量 Item 决定。也可以采用以下代码代替：

```
Cells(Item, "B").EntireRow.Delete
```

（3）在记录被删除的行号时应该采用分隔符加以区分，否则所有行号连在一起将无法识别。本例采用换行符作为分隔符，这样在 MsgBox 函数产生的信息框的显示效果将更美观。

（4）执行删除行或者插入行这类操作时会快速地更新屏幕显示效果，例如删除第 1 行后第 2 行会马上上移，从而及时地更新屏幕显示效果。但是在代码执行过程中频繁地更新屏幕显示内容会造成代码执行效率低下。对于这类操作，可以在执行循环语句前关闭屏幕更新，执行循环语句后再一次性更新，从而大幅度提升代码执行效率。

```
Sub 删除空白单元格所在行 2()  '放置位置：模块中
'定义两个变量，前者表示循环语句的计数器，后者用于存放所有已删除行的行号
    Dim Item As Integer, RowName As String
    Application.ScreenUpdating = False             '关闭屏幕更新
    '从 B 列最后一个非空单元格的行号开始，到第一行结束，步长值为-1
    For Item = Cells(Rows.Count, "B").End(xlUp).Row To 1 Step -1
        If Len(Cells(Item, "B")) = 0 Then          '如果单元格中没有字符
            Rows(Item).Delete                      '那么删除整行
            RowName = RowName & Chr(13) & Item     '记录被删除的行
        End If
    Next Item
    Application.ScreenUpdating = True              '恢复屏幕更新
    '如果变量 RowName 非空则报告已删除的行号
    If Len(RowName) > 0 Then MsgBox "以下行已成功删除：" & RowName
End Sub
```

在循环语句前关闭屏幕更新，循环语句结束后恢复屏幕更新，这是提升代码执行效率的通用手法。读者可以使用超过一万行数据来测试使用 ScreenUpdating 前后的代码执行时间比较，需要删除的行越多，效率差异越明显。

在关闭屏幕更新的状态下，将看不到代码执行过程，单元格不会实时更新，当所有操作过程结束时再将 ScreenUpdating 属性赋值为 True 即可恢复屏幕更新，相当于将多次更新压缩为只更新一次，因此可以加快代码执行速度。但是在代码执行结束时必须恢复屏幕更新，否则会影响日常工作。

3．检查工作簿中是否存在图形对象

要求:检查活动工作簿中是否存在隐藏的图形对象，如果有则记录该工作表名称。

隐藏的图形对象会增加文件的体积，消耗更多的资源。如果可以确定工作簿中不存在有用的且处于隐藏状态的图形对象，那么就有必要检查工作簿中是否存在隐藏的图形对象，从而给工作簿“减肥”。

本例需要使用双层循环，即 For Next 嵌套应用，完整代码如下。

```
Sub 判断哪些工作表中有隐藏图片()  '放置位置：模块中
    '声明两个变量,前者作为外层循环的计数器,后者作为里层循环的计数器
    Dim Sht As Integer, Shp As Integer, ShtName As String
    For Sht = 1 To Worksheets.Count  '从第一个到最后一个工作表
        For Shp = 1 To Worksheets(Sht).Shapes.Count     '从第一个到最后一个图形对象
            '如果第 Sht 个工作表的第 Shp 个图形对象处于隐藏状态
            If Worksheets(Sht).Shapes(Shp).Visible = False Then
                '将工作表名称与换行符串连起来,赋值给变量 ShtName
                ShtName = ShtName & Chr(10) & Worksheets(Sht).Name
                Exit For   '然后终止本层循环（本例中两层循环,此处只能终止里层的循环）
            End If
        Next Shp        '检查下一个图形对象
    Next Sht            '检查下一个工作表
    If Len(ShtName) > 0 Then MsgBox "以下工作表存在隐藏图形对象:" & ShtName, vbOKOnly,    "友情提
示"
End Sub
```

执行此过程后，如果工作簿中的所有工作表都没有隐藏的图形对象，那么不会有任何反应；如果工作簿中有任意工作表存在隐藏的图形对象，那么会弹出信息框，并罗列出具有隐藏图形对象的工作表名称。图 6-18 是本书案例文件的执行结果。

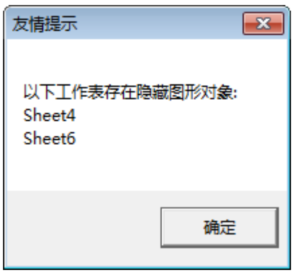


图 6–18　有隐藏图形对象的工作表名称

代码分析

（1）常用的表包括图表和工作表，一般没有必要检查图表中是否存在隐藏图片，因此本例中

外层循环的终止值是“Worksheets.Count”，而不是“Sheets.Count”，从而可以忽略图表。

（2）里层循环的终止值是指定工作表中的图形对象数量，而工作表对象是由 Sht 指定的，所以引用工作表采用“Worksheets(Sht)”，而计算该表中的图形对象数量则用以下代码：

```
Worksheets(Sht).Shapes.Count
```

（3）图形对象的 Visible 属性代表可见状态，如该属性值为 False，则表示处于隐藏状态，否则处于显示状态。

（4）使用 For Next 双层嵌套时要注意 For Next 与 Next 的对应关系。本例外层循环的计数器是 Sht，里层循环的计数器是 Shp，所以外层和里层的 Next 也需要与 Sht、Shp 对应。Next Sht 和 Next Shp 出现的顺序与 For Sht 和 For Shp 的顺序刚好相反。

（5）Exit For 用于退出本层循环，相当于程序跳转到 Next 之后的语句。它只能结束当前层的循环，本例采用了双层循环，所以当里层循环结束后外层的循环可以继续执行。

（6）如果将要求修改为判断工作簿中是否存在隐藏的图形对象，不需要罗列出所有包含图形对象的工作表名称，那么可以在遇到第一个隐藏的图形对象后就停止所有循环，Exit For 可以直接修改为 End，表示结束一切，不过也需要将 Msgbox 函数移到 End 之前。

6.5　For Each...Next 语句解析

For Each...Next 用于对象集合与数组，即针对一个对象集合或数组中的每个元素重复执行一组语句。它和 For Next 的功能和用法都相近，当学会 For Next 后稍加学习，即可掌握 For Each...Next 语句的应用思路。

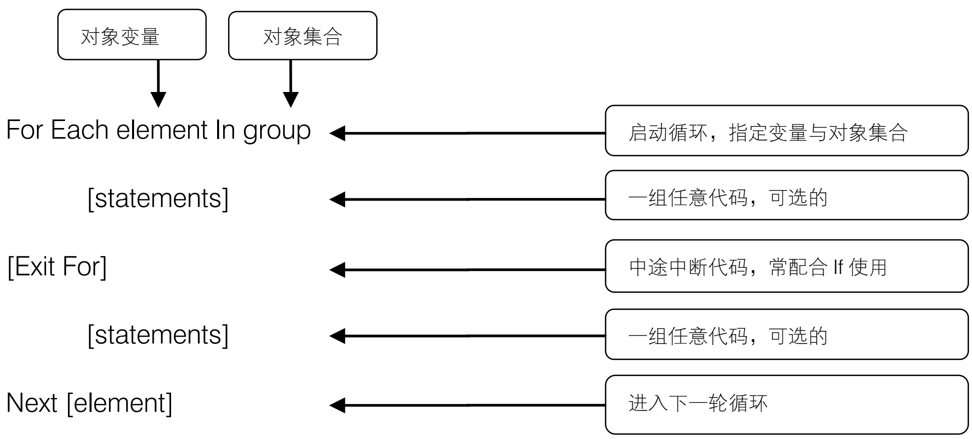
6.5.1　遍历对象集合

For Each...Next 语句用于数组和对象集合，其中更常见的应用是对象集合，例如工作表集合、工作簿集合、窗口集合、批注集合、图形对象集合或者图表集合。

换言之，操作对象是集合时可以通过对象变量遍历整个集合，然后逐一操作其中的每个子对象。不过 For Next 语句也可以实现遍历对象集合，实现 For Each...Next 循环语句的同等功能，并不强制要求使用哪一种循环语句。

6.5.2　For Each...Next 语句基本语法

For Each...Next 语句的语法和 For Next 语句的语法相近，具体为：



其中 group 表示数组或者对象集合，element 表示对象变量，通过此变量去遍历数组或者对象集合中的每一个子元素。

Statements 表示循环体中的一句或者多句代码，Exit For 用于退出循环，两者都是可选项。通常 Exit For 会配合 If 使用，即符合指定条件时终止循环。

For Each...Next 循环总是从集合中第一个元素开始，直到最后一个元素结束，不可以调整循环的方向以及步长值。

如果利用 For Each...Next 循环代替 For Next 循环，那么 6.4.4 节的过程“检查工作簿中是否存在图形对象”可以修改如下。

```
Sub 判断哪些工作表中有隐藏图片()      '放置位置：模块中
   '声明两个对象变量，前者用于遍历工作表，后者用于循环图形对象
    Dim Sht As Worksheet, Shp As Shape, ShtName As String
    For Each Sht In Worksheets             '遍历所有工作表
        For Each Shp In Sht.Shapes        '遍历 Sht 工作表中的所有图形对象
            '如果工作表 Sht 工作表中的图形 Shp 处于隐藏状态
            If Shp.Visible = False Then
                '将工作表名称与换行符串连起来,赋值给变量 ShtName
                ShtName = ShtName & Chr(10) & Sht.Name
                Exit For  '然后终止本层循环（本例中两层循环,此处只能终止里层的循环）
            End If
        Next Shp        '检查下一个图形对象
    Next Sht            '检查下一个工作表
    If Len(ShtName) > 0 Then MsgBox "以下工作表存在隐藏图形对象:" & ShtName, vbOKOnly, "友情提示"
End Sub
```

读者可以比较此过程与 6.4.4 节中使用 For Next 循环的过程有何差异，它们实现的功能相同，只是将 For Next 循环改为 For Each...Next 循环而已。

6.5.3 For Each...Next 语句应用案例：定位大于某值的单元格

For Each...Next 语句的主要功能是遍历对象集合，可以对其中每个子对象逐一执行某些操作或者判断。下面通过案例展示 For Each...Next 语句的应用思路。

要求：定位选区中大于参照值的所有单元格，参照值由用户手动指定。

实现此要求需要涉及至少 4 个知识点：第 1 个知识点是利用 Application.InputBox 方法产生一个输入框供用户指定参照值。第 2 个知识点是需要通过 If 语句判断用户当前选择的是单元格还是区域，如果选择了单元格，那么将操作对象设置为工作表的已用区域，否则将选区作为操作区域。第 3 个知识点是通过 For Each...Next 语句遍历操作区域，逐一判断其值是否符合条件。第 4 个知识点是将所有符合条件的单元格合并为一个 Range 对象，从而方便定位。

实现本例需求的完整代码如下：

```
Sub 定位大于某值的所有单元格()       '放置位置：模块中
    If TypeName(Selection) <> "Range" Then End  '如果选择对象不是单元格,那么结束程序
     '定义 Range 型的变量，用于代表操作区域，使用变量参与后续的运算可提升代码效率
    Dim 操作区域 As Range
    If Selection.Cells.Count = 1 Then      '如果只选择了单个单元格
        Set 操作区域 = ActiveSheet.UsedRange  '将变量赋值为活动工作表的已用区域
Else                                       '否则
'将变量赋值为选区与已用区域的交集
        Set 操作区域 = Intersect(ActiveSheet.UsedRange, Selection)
```

```
        End If
         '定义两个 Range 型的变量，和一个变体型变量“参照值”，该变量必须用变体型变量，否则无法
    '区别用户输入了 0 还是按下了“取消”按钮
    Dim 单元格 As Range, 目标区域 As Range, 参照值
    '让用户在输入框中输入参照值
        参照值 = Application.InputBox("请指定参照值:", "参照值", , , , , , 1)
        If 参照值 = "False" Then End                    '如果用户按下了“取消”按钮则结束程序
        For Each 单元格 In 操作区域                     '遍历“操作区域”的每一个元素
            If IsNumeric(单元格) Then                   '如果变量“单元格”的值是数值
                If 单元格.Value > 参照值 Then           '如果“单元格”的值大于“参照值”
                    '如果“目标区域”尚未初始化，那么将“单元格”赋值给“目标区域”，
    '否则将“目标区域”与“单元格”合并，再赋值给“目标区域”
                    If 目标区域 Is Nothing Then Set 目标区域 = 单元格 Else Set 目标区域 = Union(目标
区域, 单元格)
                End If
            End If
        Next 单元格
        If Not 目标区域 Is Nothing Then 目标区域.Select '如果"目标区域"已初始化则选中"目标区域"
    End Sub
```

选择任意单元格后执行以上过程将会弹出一个输入框，在里面输入参照值 60，如图 6-19 所示。当单击“确定”按钮后可以选中工作表中所有大于 60 的单元格，效果如图 6-20 所示。

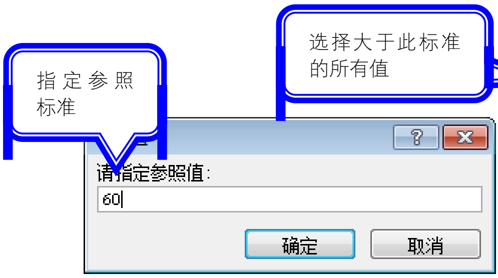
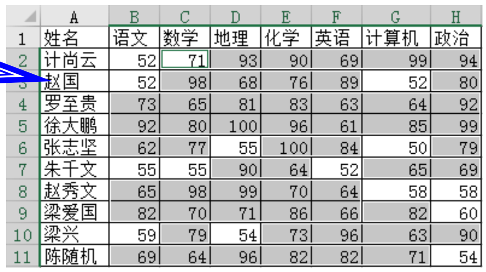


图 6–19　指定参照值



	A	B	C	D	E	F	G	H
1	姓名	语文	数学	地理	化学	英语	计算机	政治
2	计尚云	52	71	93	90	69	99	94
3	赵国	52	98	68	76	89	52	80
4	罗至贵	73	65	81	83	63	64	92
5	徐大鹏	92	80	100	96	61	85	99
6	张志坚	62	77	55	100	84	50	79
7	朱千文	55	55	90	64	52	65	69
8	赵秀文	65	98	99	70	64	58	58
9	梁爱国	82	70	71	86	66	82	60
10	梁兴	59	79	54	73	96	63	90
11	陈随机	69	64	96	82	82	71	54

图 6–20　全选已用区域中大于 60 的单元格

如果选择 D 列和 E 列后再执行以上过程，并将参照值设定为 90，那么执行结果如图 6-21 所示。



	A	B	C	D	E	F	G	H
1	姓名	语文	数学	地理	化学	英语	计算机	政治
2	计尚云	52	71	93	90	69	99	94
3	赵国	52	98	68	76	89	52	80
4	罗至贵	73	65	81	83	63	64	92
5	徐大鹏	92	80	100	96	61	85	99
6	张志坚	62	77	55	100	84	50	79
7	朱千文	55	55	90	64	52	65	69
8	赵秀文	65	98	99	70	64	58	58
9	梁爱国	82	70	71	86	66	82	60
10	梁兴	59	79	54	73	96	63	90
11	陈随机	69	64	96	82	82	71	54

图 6–21　全选 D、E 列大于 90 的单元格

经过测试，以上代码可以实现需要的一切功能，同时又拥有较好的防错性与兼容性。

代码分析

（1）所有针对单元格的操作之前都有必要检查当前选择的对象是否为单元格，如果用户选择的是图表或者其他图形对象，要么退出程序，要么提示用户选择单元格。判断当前选择的对象是

否为单元格，采用“TypeName(Selection)”即可，当返回值为 Range 时表示是单元格。

（2）Excel 的排序、筛选、查找、替换等操作皆设置为选择单个单元格时以活动单元格的当前区域或者工作表的已用区域作为操作对象，而选择多个单元格时则以选区作为操作对象。该思路比较人性化，所以此处将它借鉴过来，当用户选择单个单元格时将活动工作表的已用数据区域作为操作对象，否则将已用数据区域与选区的交集作为操作对象。

（3）对选区执行任意操作，都有必要将 Selection 重置为 Intersect(ActiveSheet.UsedRange, Selection)，从而可以忽略空白区域，节约内存资源。如果直接用 Selection 作为操作对象，那么循环语句会遍历其中每个单元格，当用户选择工作表时执行太多不必要的操作。

（4）本例中的参照值通过 Application.InputBox 方法由用户指定，所以 Application.InputBox 方法的最后一个参数需要使用 1，表示强制用户录入数值。但是这里将产生一个新的问题——当用户输入 0 或者单击“取消”按钮时都会得到 0，因此无法判断用户是否单击“取消”按钮。本例采用的解决方案是将变量定义为变体型，当用户单击“取消”按钮时其值为“False”，不再和输入 0 时的返回值相同，从而可以正确地区分它们。

（5）VBA 内部默认文本大于数值，所以本例有必要使用 If 加 IsNumeric 判断单元格中是否为数值，若是文本则忽略，否则所有文本所在单元格都会成为符合条件的对象。

（6）变量“目标区域”在赋值之前不是单元格对象，而是 Nothing，不可以通过 Union 将它与单元格对象合并，所以本例先判断该变量是否为 Nothing。如果是，则将第一次找到的目标单元格赋值给变量；如果不是，则可以通过 Union 方法执行合并。

（7）编程时要考虑一切意外情况，提升代码的兼容性。例如本例中有可能活动工作表是空表，或者所有单元格都是文本，所以在代码中需要有足够的判断语句，既提升代码执行效率，又避免因没有找到符合条件的目标导致执行 Select 操作出错。

本例除展示循环语句 For Each…Next 的应用思路外，在实际工作中也有较高实用性。因为 Excel 自身只能实现定位等于某值的所有单元格，对于定位大于、小于某个值的单元格无能为力，本例可以弥补此缺陷。

本例案例文件请参考：..\ 第 6 章\ 6-9 For Each Next 应用.xlsm

6.6 Do Loop 语句解析

Do Loop 循环语句的功能是，只要设定的条件成立就一直循环执行一组语句，或者一直循环执行一组命令，直到设定的条件成立时才会停止执行。可以通过控制条件让程序永远执行下去，或者让程序在适当的时候停止。Do Loop 循环能完成一些 For…Next 和 For each… Next 循环语句无法完成的工作。

Do Loop 包含四种用法，功能和语法皆不相同。

6.6.1 Do Loop 语法分析

Do Loop 的语法较复杂，包含四种形式，分别对应四种设置条件的方式。

其一是当条件为 True 时，重复执行一个语句块中的命令，其语法如下。

Do [While condition]	启动循环，条件（可选的）为 True 时重复执行代码
[statements]	一组任意代码，可选的
[Exit Do]	中途中断循环，常配合 If 使用，可选的
[statements]	一组任意代码，可选的
Loop	结束语句，但是否终止循环由 condition 决定

其中 While condition 是可选的，表示如果条件为 True 就执行后面的命令，直到条件不成立时结束。由于条件是可选的，所以可以忽略条件，让命令 statements 反反复复地永远循环下去。

Exit Do 表示终止本层循环，通常搭配 If 使用，表示满足某条件时终止循环。

其二是当条件为 False 时，重复执行一个语句块中的命令，其语法如下。

Do [Until condition]	启动循环，条件（可选的）为 False 时重复执行代码
[statements]	一组任意代码，可选的
[Exit Do]	中途中断循环，常配合 If 使用，可选的
[statements]	一组任意代码，可选的
Loop	结束语句，但是否终止循环由 condition 决定

同样的，条件 Until condition 和终止循环命令 Exit Do 都是可选项。

其三是执行一个语句块中的命令，如果条件为 True 就继续执行，其语法如下。

Do	启动循环
[statements]	一组任意代码，可选的
[Exit Do]	中途中断循环，常配合 If 使用，可选的
[statements]	一组任意代码，可选的
Loop [While condition]	如果条件 condition 为 True 就继续执行

要注意，这种形式的 Do Loop 语句已经将条件判断语句放在命令之后，表示先执行命令后判断条件，如果条件成立那么继续执行，否则结束循环。此时的 Do Loop 语句不管条件如何都会先执行一次命令，然后根据条件判断是否继续执行。

其四是执行一个语句块中的命令，只要条件为 False 就继续执行，其语法如下。

Do	启动循环
[statements]	一组任意代码，可选的
[Exit Do]	中途中断循环，常配合 If 使用，可选的
[statements]	一组任意代码，可选的
Loop [Until condition]	如果条件 condition 为 False 就继续执行

以上是 Do Loop 语句的四种结构形式，接下来讲解它们的应用，从而更深层次地了解 Do Loop 语句。

6.6.2 Do Loop 语法一应用

Do Loop 的语法一可以称为 Do While condition…Loop，表示只要条件成立就会一直执行一组代码。

图 6-22 展示了 10 个学生的多科目成绩，现要求通过代码按范围选择单元格，范围不能固定，须在启动代码后由用户手动指定，具体代码如下。

```
Sub 按范围定位 1()    '放置位置：模块中
  Dim 范围 As String, 下限 As Double, 上限 As Double , Bl As Boolean '声明 4 个变量
  Bl = True '预先给变量 Bl 赋值为 True
  Do While Bl '只要 Bl 的值为 True，就一直循环下去
 '弹出对话框让用户指定搜索范围
    范围 = Application.InputBox("请指定查询范围，格式为“60-80”", "指定范围", , , , , , 2)
    If Len(范围) >= 3 And InStr(范围, "-") > 0 Then'如果录入的值长度大于等于 3 个，而且包含“-”
      '如果“-”前后的值都是数值（Instr 函数计算“-”的位置，然后用 Left 函数提取它左边的字符，'用Right 提取它右边的字符）
      If IsNumeric(Left(范围, InStr(1, 范围, "-") - 1)) And IsNumeric(Right(范围, Len(范围) - InStr(1, 范围, "-"))) Then
        下限 = Left(范围, InStr(1, 范围, "-") - 1)   '将“-”左边的字符赋给值变量“下限”
        上限 = Right(范围, Len(范围) - InStr(1, 范围, "-"))   '将“-”右边的字符赋给值变量“上限”
        '如果下限小于上限，那么将变量 Bl 赋值为 False，否则提示用户录入有误，然后继续循环
        If 下限 < 上限 Then  Bl = False Else MsgBox "下限必须小于上限，请重新输入范围。", vbOKOnly, "友情提示"
      Else  '否则（"-"的前面不是数值或者后面不是数值）
        '提示用户录入有误，然后继续循环
        MsgBox "范围的上限和下限必须是数值，请重新输入范围。", vbOKOnly, "友情提示"
      End If
    Else  '否则（长度小于 3 或者没有“-”），提示用户录入有误，然后继续循环
      MsgBox "请按“60-80”这种格式录入，请重新输入范围。", vbOKOnly, "友情提示"
    End If
  Loop
  Dim cell As Range, TargetRng As Range, RngCount As Integer   '声明 3 个变量
  For Each cell In Range("B2:G11")   '遍历 Range("B2:G11")区域中每个单元格
    '如果变量 Cell 代表的单元格小于等于变量“上限”同，而且大于等于变量“下限”
    If cell.Value <= 上限 And cell.Value >= 下限 Then
```

```
            RngCount = RngCount + 1    '累加计数器（默认值等于 0）
            '如果找到第一个，则把变量 cell 赋值给 TargetRng，否则把 TargetRng 与 Cell 合并为一个 Range 对象
            If RngCount = 1 Then Set TargetRng = cell Else Set TargetRng = Union(cell, TargetRng)
        End If
    Next cell
    If RngCount > 0 Then TargetRng.Select    '如果有符合条件的单元格，那么选中所有目标单元格
End Sub
```

	A	B	C	D	E	F	G
1	姓名	语文	数学	化学	地理	政治	体育
2	古玲玲	60	84	96	55	51	61
3	陈胡明	66	61	63	83	92	72
4	罗贵生	80	94	100	64	60	98
5	吴鑫	65	75	60	66	85	57
6	赵月峨	75	50	51	61	56	70
7	罗翠花	75	74	94	71	80	87
8	胡华	98	74	99	83	93	69
9	陈览月	57	87	97	89	68	94
10	陈丽丽	76	64	90	90	75	84
11	陈冲	100	76	97	73	81	52

图 6–22　学生成绩表

在图 6-22 所示的工作表中调用过程“按范围定位 1”，然后在弹出的对话框中录入条件“三五-八十”（见图 6-23），当单击“确定”按钮后程序会弹出错误提示“范围的上限和下限必须是数值，请重新输入范围。”，效果如图 6-24 所示。

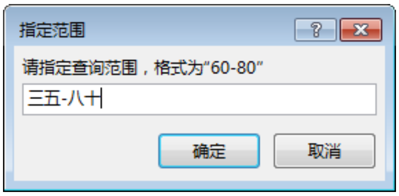


图 6–23　录入错误范围

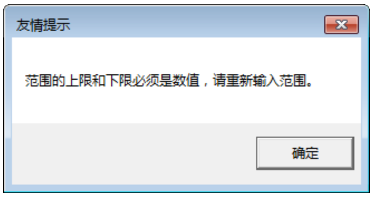


图 6–24　错误提示 1

当关闭提示信息后，程序会再次弹出输入框，此时录入范围为“60 到 90”，当单击“确定”按钮后，程序会弹出如图 6-25 所示的错误提示，然后重新弹出输入框；如果录入范围“88-60”，那么会弹出如图 6-26 所示的错误提示，接着再弹出录入框让用户重新录入范围。

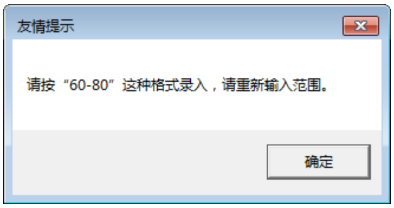


图 6–25　错误提示 2

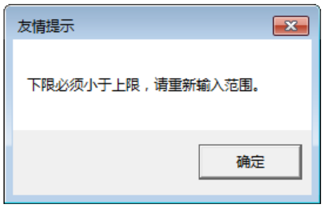


图 6–26　错误提示 3

如果录入正确的范围“70-90”，那么程序不会重新弹出错误提示，而是会选中 Range("B2:G11") 区域中大于等于 70 且小于等 90 的所有单元格。图 6-27 和图 6-28 分明对应录入条件范围和选中目标区域操作结果。

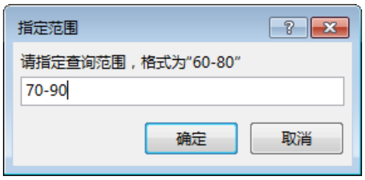


图 6–27　录入正确范围



	A	B	C	D	E	F	G
1	姓名	语文	数学	化学	地理	政治	体育
2	古玲玲	60	84	96	55	51	61
3	陈胡明	66	61	63	83	92	72
4	罗贵生	80	94	100	64	60	98
5	吴鑫	65	75	60	66	85	57
6	赵月娥	75	50	51	61	56	70
7	罗翠花	75	74	94	71	80	87
8	胡华	98	74	99	83	93	69
9	陈览月	57	87	97	89	68	94
10	陈丽丽	76	64	90	90	75	84
11	陈冲	100	76	97	73	81	52

图 6–28　选中 70–90 之间的单元格

代码分析

（1）Do While condition…Loop 表示只要条件成立就一直执行一组代码，那么在启动 Do Loop 循环之前必须让变量 Bl 的值为 True，否则 Do Loop 循环无法启动。在 Do 和 Loop 的中间再根据数据的变化来修改变量 Bl 的值，当它赋值为 False 时循环就会终止，否则会一直循环下去。就本例而言，如果用户录入的表示范围的值不符合条件，那么 Do Loop 循环就会反复启动，让用户重新录入范围；如果用户第一次录入的范围就符合条件，那么变量 Bl 被赋值为 False，Do Loop 循环会立即终止。

（2）范围的最小长度是 3，中间必须有“-”，例如“5-9”，因此在 Do Loop 循环中使用条件“Len(范围) >= 3 And InStr(范围, "-") > 0”检测用户录入的字符串，如果符合条件则对变量 Bl 赋值为 True，否则提示用户录入有误，然后进入下一轮循环。

（3）InStr 函数用于计算一个字符串在另一个字符串中的出现位置，如果出现了多次，那么只计算第一次的位置，其语法如下：

```
InStr([start, ]string1, string2[, compare])
```

第一参数代表起始搜索位置，是可选参数，默认值为 1；第二参数代表其中搜索的字符串；第三参数代表搜索目标；第四参数代表比较方式，如果搜索的是字母，那么可以通过第四参数控制是否区分大小写。

假设要在“听说长江很长”中计算“长”的首次出现位置，那么可以用以下代码：

```
MsgBox InStr(1, "听说长江很长", "长")
```

如果要搜索“长”的最后一次出现的位置，那么可以改用为以下代码：

```
MsgBox InStrRev("听说长江很长", "长")
```

要注意的是，InStrRev 没有“start”这个可选参数。

（3）在符合“Len(范围) >= 3 And InStr(范围, "-") > 0”两个条件后，还需要继续检查用户在“-”的左右录入的字符是否都是数值。例如“小-大”同时符合前面的两个条件，但是它显然不是一个正确的范围，因此需要再次通过“IsNumeric(Left(范围, InStr(1, 范围, "-") - 1)) And IsNumeric(Right(范围, Len(范围) - InStr(1, 范围, "-")))”判断“-”左边和右边的字符是不是数值。

IsNumeric 函数用于判断参数是不是数值，如果是数值则返回 True。Len 函数用于计算字符的长度，Left 函数用于提取字符串左边指定长度的字符，Right 函数用于提取字符串右边指定长度的字符。Len 经常与 Left 或者 Right 函数搭配使用。

以下代码有助于理解 Instr、Len、Right 三者的功能和搭配方式：

```
Sub 提取右边的数值()'放置位置：模块中
    数据源 = "602.5-988.9"        '指定数据源
    位置 = InStr(1, 数据源, "-")   '计算“-”的位置
    '用数据源的总长度减去“-”的位置，即为“-”右边的数字的长度，然后用 Right 函数根据此长度提取
数值，结果为 988.9
    MsgBox Right(数据源, Len(数据源) - 位置)
```

```
End Sub
```

（4）因为声明变量时，将“上限”和“下限”声明成了 Double，不能对它赋值文本，所以必须先用代码“IsNumeric(Left(范围, InStr(1, 范围, "-") - 1)) And IsNumeric(Right(范围, Len(范围) - InStr(1, 范围, "-")))”排除文本，然后使用“下限 = Left(范围, InStr(1, 范围, "-") - 1)”语句对变量赋值，否则当用户输入“三十-九十”这类文本时，程序运行会出错。

（5）当所有条件都成立时，本例采用 For Each…Next 循环语句逐一获取 Range("B2:G11") 区域中每个单元格的值，并用它们与条件范围进行比较，如果符合条件则记录下单元格，在循环结束后选中这些单元格。

（6）在记录符合条件的单元格时，不能直接用“Set TargetRng = Union(cell, TargetRng)”，而是必须采用“If RngCount = 1 Then Set TargetRng = cell Else Set TargetRng = Union(cell, TargetRng)”，因为刚声明变量后，不论是 TargetRng 还是 Nothing，都不包含具体的单元格，不能作为 Union 的参数参与合并。解决办法是在找到第一个符合条件单元格的时，使用“Set TargetRng = cell”语句将 TargetRng 转换成单元格对象，此时它包含一个具体的单元格，此后如果继续找到新的目标单元格，就可以使用 Union 方法将新的单元格对象追加到 TargetRng 中去。

基于以上分析，本例的代码也可以不用变量 RngCount，仅根据 TargetRng 是否为 Nothing 这一点就足够判断当前是否第一个找到目标。修改办法是将 Do Loop 循环后面的代码替换成以下代码。

```
    Dim cell As Range, TargetRng As Range   '声明 2 个变量
    For Each cell In Range("B2:G11")   '遍历 Range("B2:G11")区域中每个单元格
      '如果变量 Cell 代表的单元格小于等于变量“上限”，而且大于等于变量“下限”
      If cell.Value <= 上限 And cell.Value >= 下限 Then
        '如果 TargetRng 尚未初始化，把那么变量 cell 赋值给 TargetRng，否则把 TargetRng 与
'Cell 合并为一个 Range 对象
        If TargetRng Is Nothing Then Set TargetRng = cell Else Set TargetRng = Union(cell, TargetRng)
      End If
    Next cell
'如果 TargetRng 已经初始化，那么选中所有目标单元格
    If Not TargetRng Is Nothing Then TargetRng.Select
```

本例案例文件请参考：..\ 第 6 章\6-14 Do Loop 应用.xlsm

6.6.3　Do Loop 语法二应用

Do Loop 的语法二可以称为 Do Until condition…Loop，用它来实现上一个案例的同等功能可以使用以下代码：

```
Sub 按范围定位 2()   'Do Until condition…Loop 应用
    Dim 范围 As String, 下限 As Double, 上限 As Double, Bl As Boolean '声明 4 个变量
    Do Until Bl   '只要 Bl 的值为 False，就一直循环下去
     '弹出对话框让用户指定搜索范围
     范围 = Application.InputBox("请指定查询范围，格式为“60-80”", "指定范围", , , , , , 2)
     If Len(范围) >= 3 And InStr(范围, "-") > 0 Then'如果用录入的值长度大于等于 3 个，而且包含“-”
        '如果“-”前后的值都是数值（用 Instr 函数计算“-”的位置，然后用 Left 函数提取它左边的字符，
'用 Right 提取它右边的字符）
        If IsNumeric(Left(范围, InStr(1, 范围, "-") - 1)) And IsNumeric(Right(范围, Len(范围) - InStr(1, 范围, "-")))
Then
          下限 = Left(范围, InStr(1, 范围, "-") - 1)   '将“-”左边的字符赋给值变量“下限”
          上限 = Right(范围, Len(范围) - InStr(1, 范围, "-"))   '将“-”右边的字符赋给值变量“上限”
```

```
            '如果下限小于上限，那么将变量 Bl 赋值为 True，否则提示用户录入有误，然后继续循环
            If 下限 < 上限 Then Bl = True Else MsgBox "下限必须小于上限，请重新输入范围。", vbOKOnly, "
友情提示"
          Else  '否则（"-"的前面不是数值或者后面不是数值）
            '提示用户录入有误，然后继续循环
            MsgBox "范围的上限和下限必须是数值，请重新输入范围。", vbOKOnly, "友情提示"
          End If
        Else  '否则（长度小于 3 或者没有“-”），提示用户录入有误，然后继续循环
          MsgBox "请按“60-80”这种格式录入，请重新输入范围。", vbOKOnly, "友情提示"
        End If
      Loop
      Dim cell As Range, TargetRng As Range, RngCount As Integer  '声明 3 个变量
      For Each cell In Range("B2:G11")  '遍历 Range("B2:G11")区域中每个单元格
        '如果变量 cell 代表的单元格小于等于变量“上限”同，而且大于等于变量“下限”
        If cell.Value <= 上限 And cell.Value >= 下限 Then
          RngCount = RngCount + 1  '累加计数器（默认值等于 0）
          '如果找到第一个，把变量 cell 赋值给 TargetRng，否则把 TargetRng 与 cell 合并为一个 Range 对象
          If RngCount = 1 Then Set TargetRng = cell Else Set TargetRng = Union(cell, TargetRng)
        End If
      Next cell
      If RngCount > 0 Then TargetRng.Select  '如果有符合条件的单元格，那么选中所有目标单元格
    End Sub
```

Do Until condition…Loop 表示不符合条件就继续循环下去，在思路上和上一个案例类似，差异体现在以下两个地方。

其一，变量 Bl 需要以 False 参与初次运算，否则 Do Until condition…Loo 语句将不会执行循环，所以本例中删除了“Bl = True”的赋值语句，Bl 将以默认值 False 参与运算。

其二，当用户录入的范围符合所有条件时，“Bl = False”改为“Bl = True”。

6.6.4 Do Loop 语法三应用

Do Loop 的语法三可以称为 Do…Loop While condition，表示先执行命令，然后判断条件，在满足条件时继续执行。

用它来实现上一个案例的同等功能可以使用以下代码。

```
    Sub 按范围定位 3()  '为 Do…Loop While condition 应用
      Dim 范围 As String, 下限 As Double, 上限 As Double, Bl As Boolean '声明 4 个变量
      Bl = True '预先给变量 Bl 赋值为 True
      Do '开始循环
       '弹出对话框让用户指定搜索范围
        范围 = Application.InputBox("请指定查询范围，格式为“60-80”", "指定范围", , , , , , 2)
       If Len(范围) >= 3 And InStr(范围, "-") > 0 Then '如果用录入的值长度大于等于 3 个，而且包含“-”
         '如果“-”前后的值都是数值（Instr 函数计算“-”的位置，然后用 Left 函数提取它左边的字符，
    '用 Right 提取它右边的字符）
          If IsNumeric(Left(范围, InStr(1, 范围, "-") - 1)) And IsNumeric(Right(范围, Len(范围) - InStr(1, 范围, "-")))
Then
            下限 = Left(范围, InStr(1, 范围, "-") - 1)  '将“-”左边的字符赋给值变量“下限”
            上限 = Right(范围, Len(范围) - InStr(1, 范围, "-"))  '将“-”右边的字符赋给值变量“上限”
            '如果下限小于上限，那么将变量 Bl 赋值为 False，否则提示用户录入有误，然后继续循环
            If 下限 < 上限 Then Bl = False Else MsgBox "下限必须小于上限，请重新输入范围。", vbOKOnly,
"友情提示"
```

```
        Else  '否则（"-"的前面不是数值或者后面不是数值）
          '提示用户录入有误，然后继续循环
          MsgBox "范围的上限和下限必须是数值，请重新输入范围。", vbOKOnly, "友情提示"
        End If
      Else  '否则（长度小于 3 或者没有“-”），提示用户录入有误，然后继续循环
        MsgBox "请按“60-80”这种格式录入，请重新输入范围。", vbOKOnly, "友情提示"
      End If
    Loop While Bl '只要条件为 True 就继续循环下去
    Dim cell As Range, TargetRng As Range, RngCount As Integer  '声明 3 个变量
    For Each cell In Range("B2:G11")  '遍历 Range("B2:G11")区域中每个单元格
      '如果变量 cell 代表的单元格小于等于变量“上限”，而且大于等于变量“下限”
      If cell.Value <= 上限 And cell.Value >= 下限 Then
        RngCount = RngCount + 1  '累加计数器（默认值等于 0）
        '如果找到第一个，把变量 cell 赋值给 TargetRng，否则把 TargetRng 与 cell 合并为一个
  'Range 对象
        If RngCount = 1 Then Set TargetRng = cell Else Set TargetRng = Union(cell, TargetRng)
      End If
    Next cell
    If RngCount > 0 Then TargetRng.Select  '如果有符合条件的单元格，那么选中所有目标单元格
  End Sub
```

Do…Loop While condition 形式的 Do Loop 语句表示先启动循环，然后再检查变量 Bl 的值是否为 True，如果是 True 则继续循环，否则结束循环。因此在启动循环语句之前需要将变量 Bl 赋值为 True，然后当用户录入的范围正确时将它修改为 False。该语法和语法一的用法相当接近，只是判断条件的放置位置不同而已。

6.6.5　Do Loop 语法四应用

Do Loop 的语法四可以称为 Do…Loop Until condition，表示先执行命令，然后判断条件，在不满足条件时继续执行。

用它实现前一个案例的同等功能的完整代码如下。

```
  Sub 按范围定位 4()    'Do…Loop Until condition 应用
    Dim 范围 As String, 下限 As Double, 上限 As Double, Bl As Boolean '声明 4 个变量
    Do '开始循环
      '弹出对话框让用户指定搜索范围
      范围 = Application.InputBox("请指定查询范围，格式为“60-80”", "指定范围", , , , , , 2)
     If Len(范围) >= 3 And InStr(范围, "-") > 0 Then '如果录入的值长度大于等于 3 个，而且包含“-”
        '如果“-”前后的值都是数值（Instr 函数计算“-”的位置，然后用 Left 函数提取它左边的字符，
  '用 Right 提取它右边的字符）
        If IsNumeric(Left(范围, InStr(1, 范围, "-") - 1)) And IsNumeric(Right(范围, Len(范围) - InStr(1, 范围, "-")))
Then
          下限 = Left(范围, InStr(1, 范围, "-") - 1)  '将“-”左边的字符赋给值变量“下限”
          上限 = Right(范围, Len(范围) - InStr(1, 范围, "-"))  '将“-”右边的字符赋给值变量“上限”
          '如果下限小于上限，那么将变量 Bl 赋值为 True，否则提示用户录入有误，然后继续循环
          If 下限 < 上限 Then Bl = True Else MsgBox "下限必须小于上限，请重新输入范围。", vbOKOnly, "
友情提示"
        Else  '否则（“-”的前面不是数值或者后面不是数值）
          '提示用户录入有误，然后继续循环
          MsgBox "范围的上限和下限必须是数值，请重新输入范围。", vbOKOnly, "友情提示"
        End If
```

```
        Else    '否则（长度小于 3 或者没有“-”），提示用户录入有误，然后继续循环
            MsgBox "请按“60-80”这种格式录入，请重新输入范围。", vbOKOnly, "友情提示"
        End If
    Loop Until Bl '只要条件为 False 就继续循环下去
    Dim cell As Range, TargetRng As Range, RngCount As Integer   '声明 3 个变量
    For Each cell In Range("B2:G11")   '遍历 Range（"B2:G11"）区域中每个单元格
        '如果变量 cell 代表的单元格小于等于变量“上限”同，而且大于等于变量“下限”
        If cell.Value <= 上限 And cell.Value >= 下限 Then
            RngCount = RngCount + 1   '累加计数器（默认值等于 0）
            '如果找到第一个，把变量 cell 赋值给 TargetRng，否则把 TargetRng 与 cell 合并为一个
'Range 对象
            If RngCount = 1 Then Set TargetRng = cell Else Set TargetRng = Union(cell, TargetRng)
        End If
    Next cell
    If RngCount > 0 Then TargetRng.Select   '如果有符合条件的单元格，那么选中所有目标单元格
End Sub
```

比较语法一、语法二、语法三和语法四，它们的思路大致相同，差异仅体现在变量 Bl 的放置位置以及变量 Bl 的初始值不同。

读者可以反复查看以上四段代码，观察它们的差异，体会不同循环语句在实现相同功能时的基本思路。

事实上 Do Loop 还有一种形式的应用，不使用 Until 或 While，直接通过 If+Exit Do 来指定结束循环的条件。通常需要代码一直循环执行下去，只在特殊状态下才终止循环的情况下可采用这方法。下面展示一个具体应用。

要求：在移动鼠标指针时在状态栏中显示鼠标指针所在列的合计值。

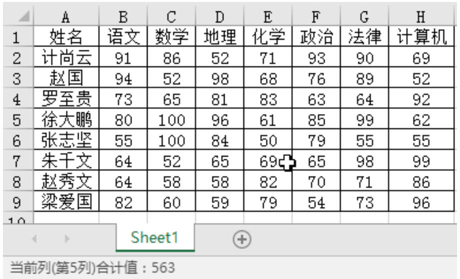
Excel VBA 本身无法调用鼠标指针的坐标值，所以鼠标指针指向哪个单元格时也无法获得该单元格的地址。不过 VBA 可以调用 API 的资源，即一些 Windows 自带的 DLL 文件中的函数，通过它们实现本不属于 Excel 的功能。

本例采用 User32.dll 文件中的 GetCursorPos 函数来获取鼠标指针的坐标值，然后通过 ActiveWindow.RangeFromPoint 方法将坐标转换成单元格对象，再调用工作表函数 Average 计算当前列的平均值，将它显示在状态栏 StatusBar 中，完整代码如下。

```
'声明 API 函数，用于获取鼠标指针移动时的坐标值
Declare Function GetCursorPos Lib "User32.dll" (lpPoint As POINTAPI) As Long
Type POINTAPI
    x As Long
    y As Long
End Type
Dim 坐标 As POINTAPI
Sub 标示当前列的合计值()
    Dim 当前单元格 As Object                '声明变量，用于代表鼠标指针下的单元格
    Do                                      '开始循环
        GetCursorPos 坐标                   '获取指针的坐标值，包括横坐标和纵坐标
'通过横坐标与纵坐标的交叉点获得对象
        Set 当前单元格 = ActiveWindow.RangeFromPoint(坐标.x, 坐标.Y)
        If TypeName(当前单元格) = "Range" Then '如果获取的对象是单元格（有可能是图形对象）
            If Range("A1").Value = "停止" Then   '如果 A1 单元格的值是“停止”
                Application.StatusBar = ""       '消除状态栏自定义的显示值
                Exit Do                          '退出循环
```

```
                Else                                    '否则
                    '在状态栏显示变量“当前单元格”所在列的合计
                    Application.StatusBar = "当前列(第" & 当前单元格.Column & "列)合计值：" &
WorksheetFunction.Sum(当前单元格.EntireColumn)
                End If
            Else                                        '否则（指获取的对象是图形对象）
                Application.StatusBar = ""              '清除状态栏自定义的显示值
            End If
            DoEvents          '转移控制权,即释放资源给其他操作,否则会耗尽资源,无法执行其他操作
        Loop
    End Sub
```

执行以上过程后，鼠标指针在工作表中移动时可在状态栏中显示鼠标指针所在列的数值合计。图 6-29 中鼠标指针在 E7 单元格，所以状态栏显示的 E 列的所有数值之和。



	A	B	C	D	E	F	G	H
1	姓名	语文	数学	地理	化学	政治	法律	计算机
2	计尚云	91	86	52	71	93	90	69
3	赵国	94	52	98	68	76	89	52
4	罗至贵	73	65	81	83	63	64	92
5	徐大鹏	80	100	96	61	85	99	62
6	张志坚	55	100	84	50	79	55	55
7	朱千文	64	52	65	69	65	98	99
8	赵秀文	64	58	58	82	70	71	86
9	梁爱国	82	60	59	79	54	73	96

图 6–29　计算鼠标指针所在列的数值之和

假设鼠标指针移到工作表之外，那么状态栏不再显示自定义信息。如果重新回到工作表区域，状态栏会自动显示新区域的数值之和。

如果在 A1 单元格录入“停止”，那么会结束循环，状态栏恢复原来的显示方式。当然，结束循环的方式很多，读者可以随意修改，例如改用双击单元格结束循环或者切换到其他工作表时结束循环，这需要使用工作表或者工作簿事件。在本书第 8 章中将详细描述如何使用事件。

代码分析

（1）Declare 用于调用 API 函数，即 Windows 自带的一些 DLL 文件中的函数，从而扩展 Excel VBA 的功能。本例中自定义类型 POINTAPI 表示鼠标指针所在的坐标，x 表示横坐标，y 表示纵坐标。代码“GetCursorPos 坐标”表示获取当前位置的鼠标指针的坐标。不同时间执行代码将取得不同的坐标。

本书重点在于 VBA 本身，不涉及 API 的教学。本节仅介绍获取鼠标指针的坐标的方法，不对 API 进行详细的讲解。

（2）Window.RangeFromPoint 方法用于获取位于屏幕上指定坐标位置的 Shape 对象或者 Range 对象。当指定位置是 Shape 对象时就返回 Shape 对象，当指定位置处是 Range 对象时则返回 Range 对象，如果指定坐标位置上没有任何形状（在 Excel 2003 中叫做自选图形），则此方法将返回 Nothing。Window.RangeFromPoint 方法的语法如下：

```
    Window.RangeFromPoint(x, y)
```

其中 x 和 y 分别代表横坐标值与纵坐标值，以像素为单位。GetCursorPos 所获得的鼠标指针位置也是以像素为单位，所以用它的值作为 Window.RangeFromPoint 方法的参数可以获得鼠标指针下的单元格对象或者图形对象。

（3）由于 Window.RangeFromPoint 方法获得的对象有可能是 Range 对象，也可能是 Shape

对象，所以变量“当前单元格”只能使用 Object，而不能采用 Range，否则当鼠标指针移到图形对象上时通过 Set 对变量赋值会产生“类型不匹配”错误。

（4）StatusBar 表示状态，可以自定义在状态栏显示的值。如果对其赋值为空文本(" ")则表示删除自定义信息，还原为默认的显示状态。

（5）如果 A1 单元格中不录入“停止”二字，那么本例中 Do Loop 会一直循环下去，直到关闭工作簿。

本例案例文件请参考：..\ 第 6 章\ 6-11 在状态栏标示当前列的合计.xlsm

6.6.6 总结三种循环语句的优缺点

常用的循环语句包含 For Next、For Each…Next、Do Loop，它们各有所长，分别适用于不同环境和需求。

◆ For Next

For Next 语句用于指定起止数值范围的循环，可以指定步长值从而改变循环体中代码的执行次数。当变量累加或者递减至终止值时自动结束循环。

中途可以随时通过 Exit For 终止循环。

◆ For Each…Next

For Each…Next 语句用于针对对象集合或者数组的循环，不可以改变步长值，即对象集合中有多少个元素，代码就会执行多少次。

中途可以随时通过 Exit For 终止循环。

For Each…Next 语句最常应用于对象集合，例如工作表对象集合 WorkSheets、单元格对象集合 Cells 或者图形对象集合 Shapes 等。

◆ Do Loop

Do Loop 循环表示只要符合条件就一直循环下去或者直到符合条件时结束循环。它没有一定的循环次数，由条件而定，有可能永远都不符合条件。

Do Loop 支持四种设置条件的形式，其条件也可以针对对象而设置，从而使 Do Loop 可以替代 For Each…Next，然而针对对象的循环采用 For Each 更易理解，Do Loop 更适合处理按未知的条件来决定循环次数的情况。

6.7 课后思考

1．如果 A1 的值小于 800，那么在 B1 单元格返回“不达标”；如果 A1 的值大于等于 800 且小于 1000，那么在 B1 单元格返回“达标”，否则返回“超额”。请分别用 If Then 语句、IIf 函数方法编写代码。

2．使用 For Next 循环和 For Each 循环两种方式实现删除已用数据区域（Worksheet.UsedRange）的偶数列的值。

3．使用 VBA 代码计算 1 到 1000 之间的个位等于 3 的所有整数之和。

4.使用 Msgbox 函数配合 Do Loop 语句实现如图 6-30 所示的信息框内容，并且在单击“否”按钮时无法关闭对话框，单击“是”按钮才能关闭。

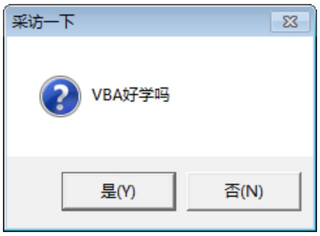


图 6–30　信息框

5．请修改“6-14 Do Loop 应用.xlsm”中的代码，不使用 While BI，也不使用 Until BI，完成同等功能。即仍然用 Do Loop 循环，但是不采用 While BI 和 Until BI 两个参数。

第 7 章　四类常见对象的应用案例

日常工作中接触最多的是单元格对象、图形对象、工作表对象和工作簿对象，而 VBA 最擅长处理的也是这些对象。

本章尽可能全面地展示这四类常见对象的常用属性和方法，通过案例学习编程思路，在学习新知识的同时巩固既有的知识，学会将它们融会贯通。

本章要点

- ◆ 单元格对象
- ◆ 图形对象
- ◆ 工作表对象
- ◆ 工作簿对象

7.1　单元格对象

单元格对象即 Range 对象，它是数据最基本的载体，在工作中使用最多的对象是单元格对象。通过本节可以熟悉单元格对象的引用方式，以及深入了解单元格对象的方法和属性，从而让工作得心应手。

7.1.1　选择单元格

【案例要求】选择 A 列最后一个非空单元格。

【知识要点】Range.Select、Range.End。

【程序代码】

```
Sub SelectRange()                                          '放置位置：模块中
 Cells(Rows.Count, "A").End(xlUp).Select        '选择 A 列最后一个非空单元格
End Sub
```

执行以上代码后可以自动定位到 A 列最后一个非空单元格，不管工作表中有多少数据。如果工作表的 A 列添加或者删除了数据，那么代码执行结果将产生相应变化。

【语法补充】

（1）Range.Select 方法用于选择活动工作表中的一个单元格或者一个区域。

（2）Range.Select 方法仅对活动工作表中的单元格生效，不能跨表选择单元格或者区域。例如活动工作表是第一个工作表，那么执行以下代码时必定出错，会提示 Select 无效。

```
Worksheets(2).Range("A2").Select
```

解决方法是将该工作表激活，然后执行 Select 动作，代码如下：

```
Worksheets(2).Select
Range("A2").Select
```

（3）VBA 提供了 Application.Goto 方法用于选择单元格或者区域，它可以跨工作表或者跨工作簿操作，如果目标单元格所在工作表或者工作簿处于未激活状态则会自动激活它，然后选择单

元格。Application.Goto 方法语法如下：

```
Application.Goto(Reference, Scroll)
```

参数 Reference 表示需要选择的目标单元格，可以带工作簿名称和工作表名称；参数 Scroll 表示是否滚动窗口，可选值为 True 和 False。当对 Scroll 赋值为 True 时表示滚动窗口直到目标区域的左上角出现在窗口的左上角中。若对 Scroll 赋值为 False 则表示仅选择目标，不滚动窗口，默认值为 False。

前面关于 Select 的问题用两句代码才能解决，而采用 Application.Goto 方法则一句足矣：

```
Application.Goto Worksheets(2).Range("a2"),True
```

不管活动工作表是哪一个工作表，执行以上代码都可以选择第 2 个工作表的 A2 单元格，且 A2 单元格处于工作表窗口的左上角。

（4）Range.Select 方法对隐藏工作表无效，Application.Goto 对隐藏工作表也无效。

本例案例文件请参考：..\ 第 7 章\ 7-1 选择单元格.xlsm

7.1.2　筛选与复制区域的值

【案例要求】将图 7-1 中产量大于等于 1000 的记录复制到 Sheet2 中。

【知识要点】Range.AutoFilter、Range.Copy。

【程序代码】

```
Sub 复制产量大于 1000 的数据到 Sheet2()                '放置位置：模块中
    With Range("B1").CurrentRegion                      '引用 B1 单元格的当前区域
    '如果活动工作表处于筛选模式，那么取消筛选（筛选状态下使用 AutoFilter 可以取消筛选，
'而非筛选状态下执行则能进入筛选状态）
        If ActiveSheet.AutoFilterMode = True Then Cells.AutoFilter
        .AutoFilter                                     '对 B1 单元格的当前区域执行筛选
        .AutoFilter Field:=2, Criteria1:=">1000"        '以第二列的值作为筛选对象，条件为大于 1000
        .Copy Sheets("Sheet2").Range("A1")              '将筛选结果复制到 Sheet2 中
        .AutoFilter                                     '复原为未筛选状态
    End With
End Sub
```

当活动工作表是 Sheet1 时执行以上代码，将得到如图 7-2 所示的结果，Sheet1 中所有大于 1000 的产量瞬间复制到 Sheet2 中。

	A	B	C
1	日期	产量	
2	11月1日	920	
3	11月2日	1180	
4	11月3日	1192	
5	11月4日	960	
6	11月5日	911	
7	11月6日	864	
8	11月8日	1059	
9	11月9日	964	
10	11月10日	965	

Sheet1　Sheet2

图 7–1　产量表

	A	B	C
1	日期	产量	
2	11月2日	1180	
3	11月3日	1192	
4	11月8日	1059	
5	11月11日	1085	
6	11月13日	1053	
7	11月16日	1033	
8			
9			
10			

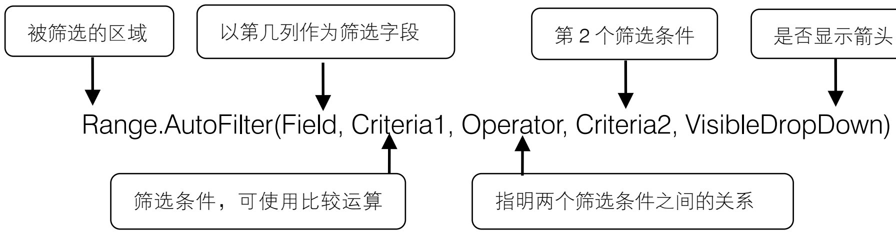
Sheet1　Sheet2

图 7–2　筛选并复制的结果

【语法补充】

（1）为了提升代码的兼容性，本例采用 Range("B1").CurrentRegion，而非固定的区域地址，从而使数据增、删时不用修改代码。

（2）Range.AutoFilter 表示对一个区域执行筛选，其语法如下：



语法中各参数的名称与含义如表 7-1 所示。

表 7-1　Range.AutoFilter的参数列表

名　称	说　明
Field	表示以第几列作为筛选字段
Criteria1	筛选条件。使用“=”可查找空字段，使用“<>”可查找非空字段。如果省略该参数，则搜索条件为所有数据
Operator	指定用于关联两个筛选条件的操作符，包括11种操作符。在帮助中搜索“XlAutoFilterOperator”即可
Criteria2	第二个筛选条件。与Criteria1和Operator一起组合成复合筛选条件
VisibleDropDown	如果为True，则显示筛选字段的自动筛选下拉箭头；如果为False，则隐藏筛选字段的自动筛选下拉箭头。默认值为True

本例中“AutoFilter Field:=2, Criteria1:=">1000"”表示以筛选区域的第二列作为筛选对象，条件为大于 1000。

如果要设置两个筛选条件，那么需要添加 Operator 和 Criteria2 参数。

（3）在筛选状态下，复制数据时会自动跳过隐藏区域，不需要通过 SpecialCells 定位可见区域后再复制。

（4） AutoFilterMode 表示工作表中是否有自动筛选的箭头，当值为 True 时表示工作表中有“自动筛选”的箭头。使用 AutoFilter 可以取消工作表的筛选箭头，再次使用 AutoFilter 则添加筛选箭头。

（5）通过 AutoFilterMode 可以判断工作表中有没有筛选箭头，而 Filter.On 属性则可以判断某一个筛选器是否指定了筛选条件。假设工作表中有 10 个筛选器（箭头），其中第 2 个筛选器使用了筛选条件，那么第一个筛选器的 Filter.On 属性值为 False,第二个筛选器的 Filter.On 属性值为 True，整个工作表的 WorkSheet.FilterMode 属性值为 True（当任何一个筛选器的使用了筛选条件时，WorkSheet.FilterMode 属性值都是 true）。

例如判断工作表是否有自动筛选的箭头或者是否设置了筛选条件，可用以下代码：

```
Sub 筛选判断()
  If ActiveSheet.AutoFilterMode Then  '如果本工作表有自动筛选的箭头
    Dim Item As Byte, FilterCount As Byte '声明两个变量
    For Item = 1 To ActiveSheet.AutoFilter.Filters.Count '遍历所有筛选器（箭头）
'记录筛选条件的数量（VBA 中 True 当作-1 参与运算，因此用 ABS 函数求它的绝对值，转换成正数）
      FilterCount = FilterCount + Abs(ActiveSheet.AutoFilter.Filters(Item).On)
    Next
    '如果设置了筛选条件，那么提示数量，否则提示未设置筛选条件
    If FilterCount Then MsgBox "本工作表设置了" & FilterCount & "个筛选条件" Else MsgBox "本工作表有筛选箭头，但未设置筛选条件", vbOKOnly, "友情提示"
  Else '否则（表示没有箭头）
    MsgBox "本工作表没使用自动筛选箭头。", vbOKOnly, "友情提示" '提示没有箭头
  End If
```

```
End Sub
```

（6）

```
Range.Copy [Destination]
```

Range 表示数据源，参数 Destination 代表目标单元格，它是可选参数，当忽略参数时表示将数据源复制到剪贴板中。

不管数据源是单元格还是区域，目标都可以只用一个单元格，VBA 会自动以它为参照点扩充区域，从而适应数据源区域的高度和宽度。

本例案例文件请参考：..\ 第 7 章\ 7-2 筛选与复制单元格.xlsm

7.1.3　多区域复制

【案例要求】Excel 本身不支持多个不规则区域的复制，即使是规则的多个区域也只能复制，粘贴后不能保持复制前的格式。本例要求对任意状态的多个区域执行复制。

【知识要点】Range.Areas 和 Range.Copy。

【程序代码】

```
Sub 多区域复制()                        '放置位置：模块中
  '声明变量 TopRow 与 LeftCol，分别用于储存行数与列数
    Dim TopRow As Long, LeftCol As Integer
    TopRow = Rows.Count                 '对变量 TopRow 赋值为最大行的行号
    LeftCol = Columns.Count             '对变量 LeftCol 赋值为最大列的列号
    Dim i As Integer                    '声明 Integer 类型的变量 i，用于循环语句中
    For i = 1 To Selection.Areas.Count  '遍历选区中的所有区域
        '如果区域的行号小于变量 TopRow，那么将新的行号赋值给变量 TopRow，
        '从而获取 Selection 的最小行号（LeftCol 也以相同方式处理）
        If Selection.Areas(i).Row < TopRow Then TopRow = Selection.Areas(i).Row
        If Selection.Areas(i).Column < LeftCol Then LeftCol = Selection.Areas(i).Column
    Next
     Dim PasteRange As Range            '声明 Range 型的变量，用于储存粘贴数据时的目标区域
'让用户指定目标单元格，程序将基于此单元格与 cells（TopRow,LeftCol）的偏移量决定如何定位目
'标区域
Set PasteRange = Application.InputBox(prompt:="请选择复制对象的存放区，如果有数据将覆盖。", Title:="选择区域", Type:=8)
  '将 Range 变量重置为其左上角的单元格（避免用户选择了区域）
     Set PasteRange = PasteRange.Cells(1)
     Application.ScreenUpdating = False      '关闭屏幕更新，从而加快代码执行速度
  '声明两个变量，分别用于储存目标区域与数据源的行差与列差
     Dim RowOffset As Long, ColOffset As Integer
  '当执行代码产生错误时，跳转到标签“错误”处（复制数据时超过边界就会出错）
     On Error GoTo 错误
     For i = 1 To Selection.Areas.Count      '遍历选区中的所有区域
        '计算当前区域的行号与变量 TopRow 的差值
         RowOffset = Selection.Areas(i).Row – TopRow
       '计算当前区域的列号与变量 LeftCol 的差值
         ColOffset = Selection.Areas(i).Column - LeftCol
       '根据两个差值确定目标区域，然后执行复制
         Selection.Areas(i).Copy PasteRange.Offset(RowOffset, ColOffset)
     Next i
     Application.ScreenUpdating = True       '恢复屏幕更新
```

```
    Exit Sub                            '退出程序，避免执行后面的代码
错误:        '设置一个标签，当前面的代码执行有误时就接着执行此处的代码
    MsgBox "超出边界，无法粘贴数据。", vbOKOnly, "友情提示"
End Sub
```

如果直接选择多个不规则的区域然后单击复制命令，那么将弹出如图 7-3 所示的错误提示。而选择图 7-3 中三个区域后执行本例的过程“多区域复制”，将弹出一个对话框让用户指定目标区域，假设选择 G3 单元格，那么现在的三个区域将被复制到以 G3 开始的区域中，且排列方式与原有的三个区域一致，如图 7-4 所示。

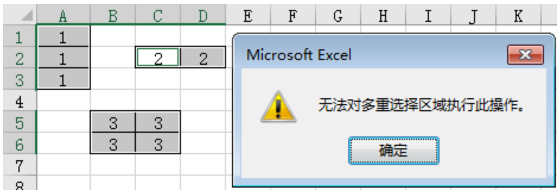


图 7–3　不允许复制不规则的多区域

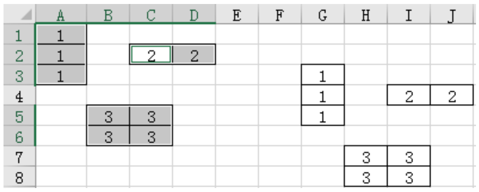


图 7–4　多区域复制结果

如果在 Application.InputBox 产生的输入框中输入了 Sheet3 中的单元格，那么 Sheet1 中的三个区域会自动被复制到 Sheet3 对应的区域中去。

在执行多区域复制时是对区域逐个执行复制的，所以如果选择的目标区域与原有的区域重叠，则可能第一次复制的数据覆盖了第二个区域的数据源，致使复制第二个数据区域时复制的数据不再是原来的数据，所以选择目标区域前应衡量复制完成后是否可能重叠，致使覆盖数据源。

【语法补充】

（1）Range.Areas 表示 Range 对象的子区域集合，Range.Areas.Count 则可以计算 Range 对象的区域个数，当选择多个区域时，Selection.Areas.Count 表示选择对象的区域数量。

（2）当选择多个区域时，Selection.Row 和 Selection.Column 只能获取第一个区域的首行行号和首列列号，所以要计算多个区域的最小行号和最小列号必须使用循环语句逐一比较，最后取最小值。

（3）当 Application.InputBox 的 Type 参数赋值为 8 时，允许用户选择一个单元格或者区域，可以跨工作表选择，但不能跨工作簿。

（4）本例重点在于计算 Application.InputBox 产生的目标单元格与选区的最小行号、最小列号之间的差异，即偏移量。Range.Copy 方法根据这个偏移量按顺序逐个复制区域。

本例案例文件请参考：..\ 第 7 章\ 7-3　多区域复制.xlsm

7.1.4　选择性粘贴数据

【案例要求】合并活动工作簿中所有工作表的数据，需要将合并对象转换成值，从而避免合并后的存放位置不同致使公式结果错误。

例如公式“=row(a1)”在 A1 单元格时结果为 1，而将它移到 B10 单元格后结果将变成 10，所以当工作表中有公式时，合并工作表应只合并数值忽略公式，但是同时有必要将公式以外的格式一起合并到新的区域中。

【知识要点】Range.PasteSpecial、Range.Copy 、Range.Borders。

【程序代码】

```
Sub 合并工作表()                        '放置位置：模块中
```

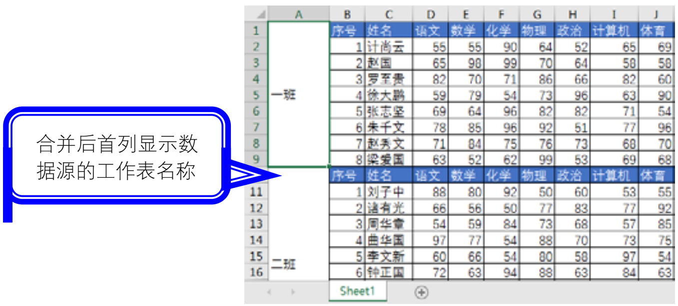
```
'声明一个 Worksheet 型变量，代表新建工作表，一个 Workbook 型的变量，用于代表活动工作簿
    Dim NewSht As Worksheet, ActiveWb As Workbook
    Set ActiveWb = ActiveWorkbook          '将活动工作簿赋值给变量 ActiveWb
    Set NewSht = Workbooks.Add.Sheets(1)  '新建一个工作簿，将它的第 1 个工作表赋值给变量
    'NewSht（此时活动工作簿不再是 ActiveWb 所代表的工作簿了）
     '声明一个 Worksheet 型变量 Sht,用于 For Each...Next 循环语句的变量，以及一个 Integer
'型的变量，作为计数器使用，代表被合并的工作表数量
      Dim Sht As Worksheet, i As Integer
      Application.ScreenUpdating = False    '关闭屏幕更新，从而提升代码执行效率
    '遍历 ActiveWb 的每一个工作表（使用 Worksheets 而不是 Sheets，会跳过图表）
     For Each Sht In ActiveWb.Worksheets
        i = i + 1                           '累加变量
        Sht.UsedRange.Copy                  '复制 Sht 工作表的已用数据区域
        '如果变量 i 的值等于 1，那么取工作表 NewSht 的 B1 赋值给变量，否则取 B 列最后一个
'非空单元格的下一个单元格赋值给变量
        Set Rng = IIf(i = 1, Range("B1"), NewSht.Cells(Rows.Count, "B").End(xlUp).Offset(1, 0))
        Rng.PasteSpecial Paste:=xlPasteFormats          '选择性粘贴格式
        Rng.PasteSpecial Paste:=xlPasteColumnWidths     '选择性粘贴列宽
        Rng.PasteSpecial Paste:=xlPasteValues           '选择性粘贴数值
        Rng.Offset(0, -1).Resize(Sht.UsedRange.Rows.Count, 1).Merge '合并首列
        Rng.Offset(0, -1) = Sht.Name                    '将原工作表名称写入合并单元格
     Next Sht
     Application.ScreenUpdating = True                  '恢复屏幕更新
     '如果变量 i 大于 0，那么将 A 列的非空单元格添加边框,且选中 A1 单元格
    If i > 0 Then Intersect(Range("a:a"), NewSht.UsedRange).Borders.LineStyle = xlContinuous:
Range("A1").Select
End Sub
```

假设工作簿中有如图 7-5 所示数据，其中 A 列采用了公式计算序号，执行过程“合并工作表”后可以得到如图 7-6 所示结果。图 7-6 是一个新建的工作簿，其中首个工作表存放合并结果，A 列用于存放数据源的工作表名称，从而便于区分数据来源。



	A	B	C	D	E	F	G	H	I
1	序号	姓名	语文	数学	化学	物理	政治	计算机	体育
2	1	计尚云	55	55	90	64	52	65	69
3	2	赵国	65	98	99	70	64	58	58
4	3	罗至贵	82	70	71	86	66	82	60
5	4	徐大鹏	59	79	54	73	96	63	90
6	5	张志坚	69	64	96	82	82	71	54
7	6	朱千文	78	85	96	92	51	77	96
8	7	赵秀文	71	84	75	76	73	68	70
9	8	梁爱国	63	52	62	99	53	69	68

图 7–5　成绩表



	A	B	C	D	E	F	G	H	I	J
1	一班	序号	姓名	语文	数学	化学	物理	政治	计算机	体育
2		1	计尚云	55	55	90	64	52	65	69
3		2	赵国	65	98	99	70	64	58	58
4		3	罗至贵	82	70	71	86	66	82	60
5		4	徐大鹏	59	79	54	73	96	63	90
6		5	张志坚	69	64	96	82	82	71	54
7		6	朱千文	78	85	96	92	51	77	96
8		7	赵秀文	71	84	75	76	73	68	70
9		8	梁爱国	63	52	62	99	53	69	68
10	二班	序号	姓名	语文	数学	化学	物理	政治	计算机	体育
11		1	刘子中	88	80	92	50	60	53	55
12		2	诸有光	66	56	50	77	83	77	92
13		3	周华章	54	59	84	73	68	57	85
14		4	曲华国	97	77	54	88	70	73	75
15		5	李文新	60	66	54	80	58	97	54
16		6	钟正国	72	63	94	88	63	84	63

图 7–6　合并结果

【语法补充】

（1）Range.PasteSpecial 表示选择性粘贴，其语法如表 7-2 所示。

```
Range.PasteSpecial(Paste, Operation, SkipBlanks, Transpose)
```

其中 Range 表示选择性粘贴的目标单元格，其四个参数的含义如表 7-2 所示。

表 7-2 PasteSpecial的参数说明

名 称	说 明
Paste	要粘贴的对象
Operation	粘贴操作
SkipBlanks	如果为True，则不将剪贴板上区域中的空白单元格粘贴到目标区域中。默认值为False
Transpose	如果为True，则在粘贴区域时转置行和列。默认值为False

其中重点在于前两个参数，后两个参数用得较少。Paste 表示选择性粘贴的对象，例如格式、公式、数值等，表 7-3 中包含了所有的可选项以及含义解释。

表 7-3 Paste参数的可选项及其说明

名 称	说 明
xlPasteAll	粘贴全部内容
xlPasteAllExceptBorders	粘贴除边框外的全部内容
xlPasteAllMergingConditionalFormats	粘贴所有内容，并且将合并条件格式
xlPasteAllUsingSourceTheme	使用源主题粘贴全部内容
xlPasteColumnWidths	粘贴复制的列宽
xlPasteComments	粘贴批注
xlPasteFormats	粘贴复制的源格式
xlPasteFormulas	粘贴公式
xlPasteFormulasAndNumberFormats	粘贴公式和数字格式
xlPasteValidation	粘贴有效性
xlPasteValues	粘贴值
xlPasteValuesAndNumberFormats	粘贴值和数字格式

Operation 参数仅在复制数值时才用，表示将复制值时采用的计算方式，包括加、减、乘、除，并且默认为不执行任何运算，如表 7-4 所示。

表 7-4 Operation参数的可选项及其说明

名 称	说 明
xlPasteSpecialOperationAdd	复制的数据与目标单元格中的值相加
xlPasteSpecialOperationDivide	复制的数据除以目标单元格中的值
xlPasteSpecialOperationMultiply	复制的数据乘以目标单元格中的值
xlPasteSpecialOperationNone	粘贴操作中不执行任何计算
xlPasteSpecialOperationSubtract	复制的数据减去目标单元格中的值

（2）本例中每个工作表的 A 列都有公式，为了避免合并后结果出错，有必要在复制数据时采用选择性粘贴的方式操作。当 Paste 参数赋值为 xlPasteValues 时表示只粘贴值，然而这显然不够人性化，通常需要将背景色和列宽一并粘贴过去，所以本例分三步执行，先粘贴格式，然后粘贴列宽，最后粘贴值。这是最人性化的一种合并方式。

（3）Range. Merge 表示合并一个区域。本例的合并对象是空白区域，所以不会产生只能保留左上角数据的提示。如果合并对象的多个数据，那么需要配合 DisplayAlerts 使用，关闭系统提示。

（4）Range. Borders 表示单元格或者区域的边框，而 Range. Borders.LineStyle 则表示边框的线型，赋值为 xlContinuous 时表示实线。线型的可选项有 8 个，如表 7-5 所示。

表 7-5　单元格边框的线型说明

名　称	说　明
xlContinuous	实线
xlDash	虚线
xlDashDot	点画相间线
xlDashDotDot	画线后跟两个点
xlDot	点式线
xlDouble	双线
xlLineStyleNone	无线条
xlSlantDashDot	倾斜的画线

（5）Application.ScreenUpdating 属性控制屏幕是否更新，当赋值为 True 时表示允许更新，否则关闭更新。所谓的屏幕更新是指 VBA 在执行过程中将每一次的执行结果都实时地显示在屏幕上，而关闭屏幕更新后则不会显示出来，它的优点是可以加快代码的执行速度。可以如此理解：在计算“=123+45+789+321+654+987”这个表达式时需要执行 5 次运算，如果每一次运算完都把答案写在纸上再执行下一次运算，那么显然比运算完 5 个步骤后再一次性将答案写在纸上慢得多。

Application.ScreenUpdating=False 一定要配合 Application.ScreenUpdating=True 使用，否则屏幕不更新执行结果，Excel 就会处于假死状态。通常在循环语句之前关闭屏幕更新，在循环结束以后再恢复更新。

本例案例文件请参考：..\ 第 7 章\ 7-4 合并所有工作表数据.xlsm

7.1.5　重置已用数据区域

【案例要求】当表格设计不合理时，已用数据区域 UsedRange 会大于实际的数据区域，即 UsedRange 的最大行号大于工作表中最后一个非空单元格的行号，或者 Usedrange 的最大列号大于工作表中最后一个非空单元格的列号。这种表格会形成“虚胖”，占用多余的系统资源，文件体积也相应增大，现要求重置已用数据区域 UsedRange 为实际的区域，从而“减肥”。

【知识要点】Range.Find、Range.Resize、Range. EntireColumn 和 Range.Delete。

【程序代码】

```
Sub 重置已用区域()      '放置位置：模块中
    '如果工作表未使用过，则退出程序
    If IsEmpty(ActiveSheet.UsedRange) Then Exit Sub
    '如果已用数据区域的最后一个单元格非空，则退出程序（表示不需要“减肥”，不存在“虚胖”问题）
    If Len(ActiveSheet.UsedRange.Cells(ActiveSheet.UsedRange.Cells.Count)) > 0 Then Exit Sub
    Dim A As Long, B As Integer   '声明变量
    '计算最后一个非空行的行号（从工作表最后一个单元格处开始按行向前查找*，结果为最后一个非空行的最右边单元格）
    A = Cells.Find(What:="*", After:=Cells(Rows.Count, Columns.Count), SearchOrder:=xlByRows, SearchDirection:=xlPrevious).Row
    '计算最后一个非空列的列号（从工作表最后一个单元格处开始按列向前查找*，结果为最后一个非空列的最下方单元格）
    B = Cells.Find(What:="*", After:=Cells(Rows.Count, Columns.Count), SearchOrder:=xlByColumns,
```

```
SearchDirection:=xlPrevious).Column
    Rows(A + 1 & ":" & Rows.Count).Delete        '删除最后一个非空行下方的所有行
    '删除最后一个非空列右方的所有列
        Cells(1, b + 1).Resize(1, Columns.Count - b - 1).EntireColumn.Delete
    End Sub
```

测试以下代码需要预先进行准备工作，否则难以明白代码的作用和思路。

假设在工作表 A1:I9 区域中有数据，然后对 A1:K12 区域设置居中对齐，此时工作表的已用数据区域是 A1:K12，而不是 A1:I9，可以通过代码“MsgBox ActiveSheet.UsedRange.Address”验证，结果如图 7-7 所示。

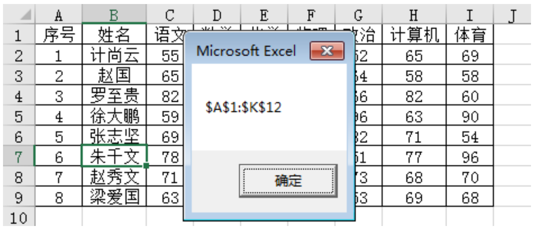


图 7–7　重置前的成绩表

对于如图 7-7 所示的数据，执行过程“重置已用数据区域”后将清除空白区域的格式，从而实现工作表“减肥”。重置后再通过代码“MsgBox ActiveSheet.UsedRange.Address”测试，可以得到地址 A1:I9。

【语法补充】

（1） ActiveSheet.UsedRange.Cells(ActiveSheet.UsedRange.Cells.Count)表示活动工作表已用区域的最后一个单元格，Cells 参数是指该区域的单元格数量。

（2）Range.Find 表示查找符合条件的单元格，其返回值为 Range 对象。Range.Find 语法如下：

```
    Range.Find(What, After, LookIn, LookAt, SearchOrder, SearchDirection, MatchCase, MatchByte, SearchFormat)
```

Range 表示查找的区域，如果 Range 是单个单元格，那么将在工作表的所有单元格中查找；如果 Range 是区域，那么将在区域中查找。

Range.Find 的语法中各部分的含义如表 7-6 所示。

表 7-6　Range.Find的参数说明

名　称	说　明
What	要搜索的数据，可以使用通配符
After	表示参照单元格，搜索过程将从此单元格之后开始执行搜索。搜索过程从该单元格之后开始，直到绕回到此单元格时，才对其进行搜索。如果不指定该参数，那么将从区域的左上角开始
LookIn	表示查找的类型，包括按批注查找（xlComments）、按公式查找（xlFormulas）、按值查找（xlValues）
LookAt	匹配方式，包括完全匹配（xlWhole）和部分匹配（xlPart）
SearchOrder	行列方式，包括按行搜索（xlByRows）和按列搜索（xlByColumns）
SearchDirection	搜索的方向，包括下一个（xlNext）和上一个（xlPrevious）
MatchCase	是否区分大小写，如果为True则搜索区分大小写。默认值为 False
MatchByte	只在已经选择或安装了双字节语言支持时适用。如果为True，则双字节字符只与双字节字符匹配
SearchFormat	搜索的格式。按格式查找时需要设置此参数

Range.Find 的 LookAt 参数有记忆功能，会记住上次的设置，这本是人性化的一个功能，但是有时会带来副作用。例如上次查找时使用了完全匹配，那么本次查找也会采用完全匹配，如果

实际需求是部分匹配则无法达到预期效果。为了解决这个问题，每次使用 Range.Find 都需要对 LookAt 参数赋值。

当有合并单元格时，Range.Find 的查找区域需要采用整个区域，否则无法查找成功。例如图 7-8 中 A 列与 B 列合并，那么通过以下代码获取 A 列中“成都”所在单元格将会出错：

```
MsgBox Range("A:A").Find("成都", , , xlWhole).Address
```

如果改用 A:B 区域作为查找区域就没有问题，以下代码的结果为“$A$2”：

```
MsgBox Range("A:B").Find("成都", , , xlWhole).Address
```

	A	B	C	D
1	上海		上海	
2	成都		成都	
3	重庆		重庆	
4	上海		上海	

图 7-8　在合并区域中查找

（3）代码“Cells.Find(What:="*", After:=Cells(Rows.Count, Columns.Count), SearchOrder:=xlByRows, SearchDirection:=xlPrevious)”表示在所有单元格中查找通配符“*”，以工作表中最后一个单元格为参照点，按行查找前一个符合条件的值。可以理解为从工作表最后一行向上找到的第一个非空行的最右边的单元格，该单元格所在行即为最后一个非空行。删除该行后面的所有行即可实现工作表“减肥”。

（4） Rows 表示对象集合，参数由数字或者数字加冒号组成。所以代码“Rows(A + 1 & ":" & Rows.Count)”代表 A+1 行以后的所有行。但是 Columns 的参数却不能使用“B+ 1 & ":" & Columns.Count”，因为列标只能是字母而非数值，所以本例删除列时改用另外的思路，先计算出需要删除的第一个单元格，然后通过 Resize 重置区域。

（5）Range.Delete 表示删除单元格，它会将整个单元格都删除。而 Range.ClearContents 表示只清除内容，Range.ClearFormats 表示仅清除格式，Range.Clear 表示清除内容和格式。只有 Range.Delete 清除最彻底，将单元格本身一并清除了。

（6）本例可对活动工作表重置已用数据区域，加上循环可以对所有工作表重置已用区域，读者可自行练习。

本例案例文件请参考：..\ 第 7 章\ 7-5　重置已用区域.xlsm

7.1.6　查找所有成绩为 100 的单元格

【案例要求】Excel 本身可以将所有查找目标选中，不过无法通过录制宏取得代码，只能手动编写。现要求查找成绩表中成绩为 100 的单元格，并将它们一次性选中。

【知识要点】Range.Find、Range.FindNext 和 Range.Address。

【程序代码】

```
Sub 查找成绩为 100 的所有单元格()                '放置位置：模块中
  '声明一个 Range 变量,用于代表 Range.Find 方法查找的目标单元格
   Dim Cell As Range
   Set Cell = Cells.Find(100, , , xlWhole)        '以完全匹配方式查找 100
   If Not Cell Is Nothing Then                    '如果已经查找到
       Dim Goal As Range, firstAdd As String      '声明一个 Range 变量和一个 String 变量
       firstAdd = Cell.Address                    '记录查找到的单元格的地址
       Set Goal = Cell                            '将目标赋值给变量 Goal
```

```
            Do                                      '开始循环
                Set Cell = Cells.FindNext(Cell)     '查找下一个
                Set Goal = Union(Goal, Cell)        '将变量与找到的下一个目标合并
    '直到下一个找到的目标和第一次找到的不相同才结束循环
            Loop While Cell.Address <> firstAdd
            Goal.Select                             '选择所有符合条件的目标区域
        Else                                        '否则
            MsgBox "不存在成绩为 100 的单元格", vbOKOnly, "友情提示"   '提示未找到目标
        End If
    End Sub
```

执行以上代码后，查找结果如图 7-9 所示。

	A	B	C	D	E	F	G	H
1	姓名	成绩	姓名	成绩	姓名	成绩	姓名	成绩
2	计尚云	55	刘子中	70	龚月新	60	曹莽	96
3	赵国	55	诸有光	64	仇正风	59	童怀礼	82
4	罗至贵	90	周华章	100	郑丽	79	罗传志	82
5	徐大鹏	100	曲华国	58	柳洪文	54	黄土尚	100
6	张志坚	52	李文新	82	魏郑	73	龙度溪	54
7	朱千文	65	钟正国	70	赵冰冰	96	梁文兴	78
8	赵秀文	69	陈强生	71	柳星华	63	陈星望	85
9	梁爱国	65	刘文喜	100	谢有金	100	穆容秋	96
10	梁兴	98	梁今明	66	林至文	69	张庆	100
11	陈随机	99	柳三秀	82	刘越堂	64	李湖云	51

图 7–9　查找结果

【语法补充】

（1）Range.Find 只能查找一个目标，而 Range. FindNext 则用于查找下一个目标。所以当需要批量查找时，先使用前者找第一个，然后用后者配合 Do Loop 语句循环查找其他所有符合条件的目标。由于 Range. FindNext 总是一直查找下一个目标，所以需要在 Do Loop 语句中限定停止查找的条件，否则会进入死循环。本例中采用“Loop While Cell.Address <> firstAdd”表示只要下一个目标的地址不等于第一次找到的目标的地址，就继续查找，事实上也可以采用“Loop Until Cell.Address = firstAdd”，两者的功能相同。

（2） FindNext 的参数是一个参照单元格，表示从该单元格之后开始查找，所以只能使用上一次找到的单元格作参数，如果使用固定的单元格则永远都只能找到同一个的目标。

（3）本例需要先查找后比较（比较找到的下一个目标是否和第一次找到的目标相同），所以只能使用 Do …Loop While 或者 Do …Loop Until，不能使用 Do While…Loop 或者 Do Until…Loop。

（4）代码“Cells.Find(100, , , xlWhole)”的结果只有两种情况：要么没找到，返回 Nothing；要么返回找到的目标单元格，返回值是一个 Range 对象，所以将它赋值给变量时需要使用 Set 语句。

本例案例文件请参考：..\ 第 7 章\ 7-6 查找所有成绩为 100 的单元格.xlsm

7.1.7　将表示平方米和立方米后面的 2 和 3 设为上标

【案例要求】将某个字符设置为上标比较简单，但是将所有单元格中的某个字符都设置为上标则比较难，若该字符与另一个字符同时出现时才标示上标，其他情况下忽略就更难了。实现此类需求只能依靠 VBA，手动操作事倍功半。本例要求将区域中“M”后面的 2 或者 3 标示为上标，其他情况下的 2 和 3 保持不变。

【知识要点】Range.Characters 和 Range.FindNext。

【程序代码】

```
Sub 将M后面的2或3标示为上标()              '放置位置：模块中
    Dim Rng As Range, i As Integer, First As String
    Application.ScreenUpdating = False     '关闭屏幕更新
    Set Rng = Cells.Find("M2", LookAt:=xlPart, LookIn:= xlValues)  '部分匹配方式查找M2
    If Not Rng Is Nothing Then             '如果找到目标
        First = Rng.Address                '记录首个符合条件的单元格的地址
        Do                                 '开始循环
            For i = 2 To Len(Rng)          '循环检查每一个字符，从第二位开始
                '如果某字符等于2且前一位是“M”
                If Mid$(Rng, i, 1) = "2" And Mid$(Rng, i - 1, 1) = "M" Then
                  '将该字符的Superscript属性设置为True，表示显示为上标
                   Rng.Characters(Start:=i, Length:=1).Font.Superscript = True
                End If
            Next
            Set Rng = Cells.FindNext(Rng)  '查找下一个
       '直到下一个找到的目标与第一个找到的单元格相同时才停止循环
        Loop Until Rng.Address = First
    End If
    Set Rng = Cells.Find("M3", LookAt:=xlPart, LookIn:= xlValues)  '部分匹配方式查找M3
    If Not Rng Is Nothing Then             '如果找到目标
        First = Rng.Address                '记录首个符合条件的单元格的地址
        Do                                 '开始循环
            For i = 2 To Len(Rng)          '循环检查每一个字符，从第二位开始
                '如果某字符等于3且前一位是“M”
                If Mid$(Rng, i, 1) = "3" And Mid$(Rng, i - 1, 1) = "M" Then
                    '将该字符的Superscript属性设置为True，表示显示为上标
                    Rng.Characters(Start:=i, Length:=1).Font.Superscript = True
                End If
            Next
            Set Rng = Cells.FindNext(Rng)  '查找下一个
        '直到下一个找到的目标与第一个找到的单元格相同时才停止循环
        Loop Until Rng.Address = First
    End If
    Application.ScreenUpdating = True      '恢复屏幕更新
End Sub
```

假设工作表中有如图 7-10 所示数据，执行以上代码后将得到如图 7-11 所示效果，其中 M2 和 M3 中的 2 和 3 显示为上标，而其他地方的 2 和 3 则保持不变。

	A
1	2012M2
2	0.38M3
3	500M2+780M2
4	7003M3
5	

图 7-10　数据源

	A
1	$2012M^2$
2	$0.38M^3$
3	$500M^2+780M^2$
4	$7003M^3$

只将 M 后面的 2 和 3 标示为上标

图 7-11　标示结果

【语法补充】

（1）本例和 7.1.6 节中的案例一样需要使用 Range.Find 加 Do Loop 循环逐一查找符合条件

的所有单元格，不过本例只需要单元格包含 M2 或者 M3，所以 LookAt 参数采用 xlPart 而非 xlWhole。

（2）VBA 未提供将单元格中指定字符标示为上标的功能，只能做到从某个位置开始的几位字符标示为上标。所以本例需要使用 Range.Characters 属性，它表示单元格内部指定长度的字符，例如“Characters(Start:=2, Length:=3)”表示第 2 位开始的 3 个字符。

（3）由于需要标示的字符 2 和 3 出现的位置只能在第 2 位到最后一位，所以本例 For Next 循环语句的初始值为 2，终止值为单元格的字符长度。

（4）Range.Find 一次只能查询一个条件，不支持 Or 运算符，因此查找 M2 和 M3 需要执行两个 Do Loop 循环。

（5）逐个调整单元格的字符时屏幕会不停地更新，影响代码执行效率。所以需要在执行循环前将 ScreenUpdating 赋值为 False，在循环结束以后再将 ScreenUpdating 恢复为 True。

本例案例文件请参考：..\ 第 7 章\7-7 将表示平方米和立方米后面的 2 和 3 设为上标.xlsm

7.1.8 合并相邻且相同的单元格

【案例要求】将一列中相同值且相邻的单元格合并，而且允许取消合并后可以还原为合并前的状态，即保留合并前的所有值。

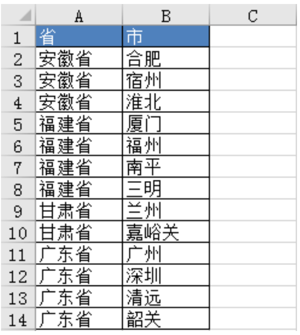
【知识要点】Range.Count、Range. Merge、Range.Copy 和 Range. PasteSpecial。

【程序代码】

```
Sub 合并列中相同数据()
  '如果选区只有一个单元格，那么提示用户，然后退出程序
  If Selection.Count = 1 Then MsgBox "请选择一个较大区域！", vbOKOnly, "友情提示": Exit Sub
  '如果选区超过 1 列则提示用户，然后退出程序
  If Selection.Columns.Count > 1 Then MsgBox "只对单列数据进行操作！", vbOKOnly, "友情提示": Exit Sub
  Application.DisplayAlerts = False            '禁止提示
  Application.ScreenUpdating = False           '关闭屏幕更新
  '声明变量
  Dim Rng As Range, Rg As Range, Rngs As Range, EndRng As Range, RngCount As Integer
  '提取选区与已用数据区域的交集并赋值给变量，从而避免执行不必要的循环，降低效率
  Set Rngs = Intersect(ActiveSheet.UsedRange, Selection)
  Set Rg = Rngs.Cells(1)                       '将区域 Rngs 的第一个单元格赋值给变量 Rg
  Set EndRng = Cells(1, Columns.Count)
  '在选区向下偏移一行的区域中循环
  For Each Rng In Rngs.Offset(1, 0)
      RngCount = RngCount + 1                  '累加计数器
    '如果计数器的值等于 Rngs 区域的单元格数量（也就是循到最后一个单元格时）
      If RngCount = Rngs.Count Then
     '在工作表最右一列创建一个辅助区，区域的高度等于需要合并的区域的高度，宽度为 1
      With EndRng.Resize(Range(Rg, Rng.Offset(-1, 0)).Rows.Count, 1)
       Range(Rg, Rng.Offset(-1, 0)).Copy EndRng  '将需要合并的区域复制到辅助区域中
       .Merge                                  '合并辅助区域
       .Copy                                   '复制辅助区域
       '将辅助区域的格式粘贴到需要合并的区域
        Range(Rg, Rng.Offset(-1, 0)).PasteSpecial xlPasteFormats
       .Clear                                  '清除辅助区域
     End With
    Else
     If Rng <> Rng.Offset(-1, 0) Then           '如果单元格 Rng 与其上一个单元不相等
```

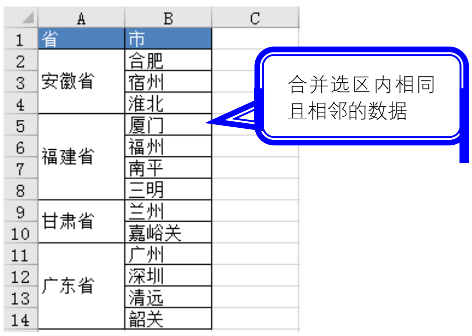
```
                '在工作表最右一列创建一个辅助区，区域的高度等于需要合并的区域的高度，宽度为 1
                With EndRng.Resize(Range(Rg, Rng.Offset(-1, 0)).Rows.Count, 1)
                  Range(Rg, Rng.Offset(-1, 0)).Copy EndRng  '将需要合并的区域复制到辅助区域中
                  .Merge                       '合并辅助区域
                  .Copy                        '复制辅助区域
                 '将辅助区域的格式粘贴到需要合并的区域
                  Range(Rg, Rng.Offset(-1, 0)).PasteSpecial xlPasteFormats
                  .Clear                       '清除辅助区域
                End With
                Set rg = Rng                   '然后重新指定对象变量（Rg 更新为 Rng 所代表的单元格）
              End If
            End If
          Next
          Application.DisplayAlerts = True    '还原提示
          Application.ScreenUpdating = True   '恢复屏幕更新
          rngs(1).Select                       '选择原选区中第一个单元格
        End Sub
```

在图 7-12 中 A 列存在许多相邻且相同的单元格，当选择 A 列后再执行过程“合并相邻且相同的单元格”，将得到如图 7-13 所示效果。



	A	B	C
1	省	市	
2	安徽省	合肥	
3	安徽省	宿州	
4	安徽省	淮北	
5	福建省	厦门	
6	福建省	福州	
7	福建省	南平	
8	福建省	三明	
9	甘肃省	兰州	
10	甘肃省	嘉峪关	
11	广东省	广州	
12	广东省	深圳	
13	广东省	清远	
14	广东省	韶关	

图 7–12 合并前效果



	A	B	C
1	省	市	
2		合肥	
3	安徽省	宿州	
4		淮北	
5		厦门	
6	福建省	福州	
7		南平	
8		三明	
9	甘肃省	兰州	
10		嘉峪关	
11		广州	
12	广东省	深圳	
13		清远	
14		韶关	

图 7–13 合并后效果

图 7-13 中相同且相邻的单元格被 VBA 代码批量合并了，其效果与手动逐一合并的一致，然而这只是外观一致，内在有较大的不同——合并后保留了所有数据。换言之，选择 A 列并单击【开始】选项卡中的【合并后居中】按钮可以还原为合并前的状态，保留一切数据，而不是合并单元格的左上角才有数据，其他单元格空白。

【语法补充】

（1）Range.Count 表示区域的单元格个数。本例中利用 Selection.Count 计算选区的单元格个数，如果只选择了单个单元格则退出程序，此状态下没有必要执行合并。

（2）本例代码仅对单列有效，在过程的前面需要检查用户选择的区域是否大于 1 列，Selection.Columns.Count 表示选区的列数，不可写作 Selection.Column。

（3）在合并有数据的多个单元格时总会弹出提示框，从而影响程序的执行效率，所以本例在合并语句之前通过代码“Application.DisplayAlerts = False”关闭提示，在过程结尾处再恢复提示。它和关闭 ScreenUpdating 属性一样，都用于代码提速。

（4）Excel 在合并区域时总是只保留第一个非空单元格的值，所以取消合并后不可能还原合并前的所有数据，这给工作带来不便。但是 Excel 却又有另一个特性——将合并单元格的格式粘贴到非合并区域中时，可以让非合并区域在外观上实现合并单元格的状态，但不会删除任何单元

格的值，所以本例利用这个原理先创建一个辅助区域，在辅助区域中执行合并，然后将辅助区域的格式复制到需要合并的区域中，从而实现对目标区域执行合并却不删除其中任意单元格的值，当对该区域取消合并后可以还原合并前的状态。

（5）为了避免浪费资源，操作对象限定为“Intersect(ActiveSheet.UsedRange, Selection)”，而不是 Selection，所以当用户选择整列时，并非整列都参与合并，只是该列与已用数据区域的交集处参与合并。

（6）循环中的判断语句“Rng <> Rng.Offset(-1, 0)”表示用单元格对象 Rng 与它上方一个单元格执行比较。如果两者相同则继续执行比较，如果两者不相同，那么 Rg 开始到 Rng.Offset(-1, 0)结束的区域就是需要合并的区域。

本例案例文件请参考：..\ 第 7 章\ 7-8 合并同类项.xlsm

7.1.9 按行合并且保留所有数据

【案例要求】对一个区域按行合并，即每行合并为一个单元格，并让用户确定是否保存所有数据。

【知识要点】Range.Rows、Range.Merge、Range.Columns 和 Range.Cells。

【程序代码】

```
Sub 按行合并()                                          '放置位置：模块中
    If TypeName(Selection) <> "Range" Then Exit Sub     '如果选区不是 Range 对象就退出程序
    If Selection.Columns.Count = 1 Then Exit Sub        '如果选区只有一列就退出程序
    Dim i As Long, j As Integer, MergeStr As String     '声明变量
    Dim Rng As Range
    Set Rng = Intersect(Selection, ActiveSheet.UsedRange) '将选区与已用区域的交集赋值给变量
    Application.DisplayAlerts = False                   '关闭提示
    '弹出对话框,如果选择是则按行合并单元格,只保留左边第一个值;否则合并时保留所有值
    If MsgBox("选是：只合并单元格" & Chr(13) & "选否：合并单元格与字符", vbYesNo, "合并方式") = vbYes
Then
        Selection.Merge True                            '执行合并（调用内置的跨越合并）
    Else                                                '否则
        For i = 1 To Rng.Rows.Count                     '从 1 到 Rng 的最后一行
            For j = 1 To Rng.Columns.Count              '从 1 到 Rng 的最后一列
                MergeStr = MergeStr & Rng.Cells(i, j).Value  '合并同行的值
            Next j
             Rng.Cells(i, 1) = MergeStr                 '将合并后的值写入最左列中
            MergeStr = ""                               '将变量清空,避免下一次使用时产生多余字符
        Next i
         Rng.Merge True                                 '执行跨越合并
    End If
    Application.DisplayAlerts = True                    '打开提示
    Selection.HorizontalAlignment = xlCenter            '让选区居中对齐
End Sub
```

假设需要对如图 7-14 所示数据执行合并，那么选择 A1:B10 区域后执行过程“按行合并”，将弹出图如 7-15 所示的对话框，在对话框中单击“是”或“否”按钮都将执行合并，不过合并方式不同。



图 7–14　合并前

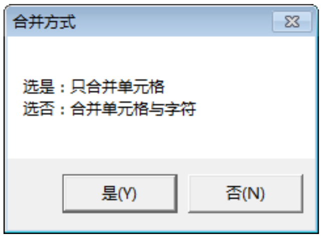


图 7–15　选择合并方式

假设单击“是”按钮，表示按行合并数据，只保留最左方的数据，合并效果如图 7-16 所示；假设单击“否”按钮，表示合并时保留行中所有数据，合并效果如图 7-17 所示。

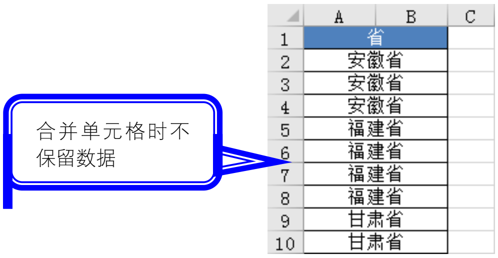


图 7–16　按行合并不保留数据



图 7–17　按行合并保留数据

【语法补充】

（1）Range.Merge 表示合并区域，对参数赋予不同值时其合并方式也不同。语法如下：

```
Range.Merge([Across])
```

Range 表示待合并的对象，必须大于 1 个单元格才有意义。合并对象允许是多个区域，程序会逐个区域执行合并。参数 Across 表示合并方式，若对参数赋值为 True，则表示将区域中每一行的单元格合并为一个单独的合并单元格；如果对参数赋值为 False，那么表示合并选区，只保留第一个非空单元格的值；如果选区有多个区域则合并为多个合并单元格。参数 Across 默认值为 False，所以忽略参数或者赋值为 False 时都表示普通的合并，当参数赋值为 True 时表示跨越合并。

（2）跨越合并时每合并一行就会弹出一个提示框，所以必须使用“Application.DisplayAlerts = False”关闭提示，在程序末尾处再恢复提示。

（3）合并单元格且保留数据时需要使用双层循环，逐个合并数据，最后一次性合并单元格，所以对 Selection 直接操作不适合，而是应当将操作对象限定为选区与已用区域的交集 Intersect(Selection, ActiveSheet.UsedRange)。

（4）在 Range.Merge 的参数为 False 或者忽略参数时表示合并后居中，而将参数赋值为 True 时表示跨越合并，没有居中功能，所以本例在过程末尾处补充了一句，将选区的 HorizontalAlignment 属性设置为 True。

本例案例文件请参考：..\ 第 7 章\ 7-9 按行合并且保留所有数据.xlsm

7.1.10　隔行插入行

【案例要求】对选区隔一行插入一个空行。

【知识要点】Range.Insert 和 Range.Resize。

【程序代码】

```
Sub 隔行插入行()                                          '放置位置：模块中
    If TypeName(Selection) <> "Range" Then Exit Sub       '如果选区不是 Range 对象就退出程序
    If Selection.Rows.Count <= 2 Then Exit Sub            '如果选区小于两行就退出程序
  '声明 Byte 型的变量,用于储存行数,可以根据需要随意设置行数
    Dim RowNum As Byte
    RowNum = 1                                            '行数预设为 1,可以随意修改
    Application.ScreenUpdating = False                    '关闭屏幕更新
  '声明变量,其中 Item 用于循环语句，Rng 用于储存已用区域与选区的交集
    Dim Item As Integer, Rng As Range
  '将选区与活动工作表的已用区域设置的交集赋予变量
    Set Rng = Intersect(Selection, ActiveSheet.UsedRange)
    For Item = Rng.Rows.Count To 2 Step -1                '开始倒序循环，即从后面向前循环
        Rng.Rows(Item).Resize(RowNum, 1).EntireRow.Insert '插入 RowNum 行
    Next Item
    Application.ScreenUpdating = True                     '恢复屏幕更新
End Sub
```

假设工作表中有如图 7-18 所示数据，选择 A2:A11 区域后执行过程“隔行插入行”，那么执行结果如图 7-19 所示，选区中除第一行外每行前都插入一个空行。



	A	B	C
1	姓名	组别	产量
2	计尚云	A组	306
3	赵国	A组	362
4	罗至贵	A组	430
5	徐大鹏	B组	419
6	张志坚	B组	356
7	朱千文	C组	445
8	赵秀文	C组	431
9	梁爱国	C组	308
10	梁兴	C组	443
11	陈随机	C组	354

图 7–18　产量表



	A	B	C
1	姓名	组别	产量
2	计尚云	A组	306
3			
4	赵国	A组	362
5			
6	罗至贵	A组	430
7			
8	徐大鹏	B组	419
9			
10	张志坚	B组	356
11			
12	朱千文	C组	445

图 7–19　隔一行插入一行

如果代码中变量 RowNum 赋值为 2，那么将会对选区隔一行插入两个空行。

【语法补充】

（1）如果只插入一行，那么插入行的代码改用“Rng.Rows(Item).EntireRow.Insert”即可，即对单元格插入整行。而本例为了提升代码的灵活性，加入了变量 RowNum，由它来决定插入多少行，用户需要插入几行就赋值为几，所以代码中需要使用 Resize，将单元格重置为高度为 RowNum 的区域，然后插入行。

（2）Range.Insert 表示插入单元格，如果 Range 是单个单元格，那么 Insert 将插入一个单元格，本例需要插入整行，所以对象 Range 需要通过 EntireRow 重置为行后再执行插入操作。

（3）由于插入新行后会影响后面的单元格的位置，所以循环语句必须从后向前循环，For Next 的起始值大于终止值，Step 采用–1，否则无法得到预期结果。

本例案例文件请参考：..\ 第 7 章\ 7-10 插行插入行.xlsm

7.1.11　标示选区中的重复值

【案例要求】将选区中重复出现的值用红圈标示出来，从而判断录入时是否存在错误。

【知识要点】Range.Validation 和 Range.Address。

【程序代码】

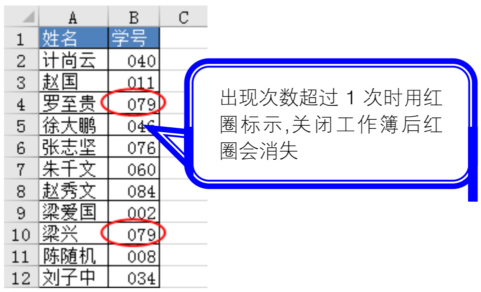
```
Sub 标示选区中的重复值()                                   '放置位置：模块中
    If TypeName(Selection) <> "Range" Then Exit Sub      '如果选区不是 Range 对象就退出程序
    If Selection.Areas.Count > 1 Then Exit Sub           '如果选区的区域个数大于 1 就退出程序
  '将选区与活动工作表的已用区域的交集赋予变量
    Set Rng = Intersect(Selection, ActiveSheet.UsedRange)
    With Rng.Validation                                  '引用 Rng 区域的有效性
        .Delete                                          '删除原有的有效性（假设有的话）
        '添加新的有效性:公式为=countif（区域地址，区域中第一个单元格地址）=1
        '公式的含义是单元格的值在区域中只出现一次才有效,否则无效
        .Add Type:=xlValidateCustom, Formula1:="=Countif(" & Rng.Address & "," & Rng.Cells(1).Address(0,
0) & ")=1"
    End With
    ActiveSheet.CircleInvalid                            '标示无效值（即重复出现的值用红圈标示）
End Sub
```

在图 7-20 中，B 列的学号具有唯一性，如果出现重复值那么表示录入有误。如果选择 B 列然后执行过程“标示选区中的重复值”，那么将得到如图 7-21 所示的效果。



	A	B	C
1	姓名	学号	
2	计尚云	040	
3	赵国	011	
4	罗至贵	079	
5	徐大鹏	046	
6	张志坚	076	
7	朱千文	060	
8	赵秀文	084	
9	梁爱国	002	
10	梁兴	079	
11	陈随机	008	
12	刘子中	034	

图 7–20　学员信息表



	A	B	C
1	姓名	学号	
2	计尚云	040	
3	赵国	011	
4	罗至贵	079	
5	徐大鹏	046	
6	张志坚	076	
7	朱千文	060	
8	赵秀文	084	
9	梁爱国	002	
10	梁兴	079	
11	陈随机	008	
12	刘子中	034	

图 7–21　标示重复值

【语法补充】

（1）Range.Validation.Add 表示对单元格或区域设置有效性验证，语法如下：

```
Range.Validation.Add (Type, AlertStyle, Operator, Formula1, Formula2)
```

其中各参数的含义如表 7-7 所示。

表 7-7　Range.Validation.Add的参数列表

名　　称	说　　明
Type	表示有效性验证的类型。详情请参阅表 7-8
AlertStyle	有效性验证警告的样式。可为以下 XlDVAlertStyle 常量之一：xlValidAlertInformation、xlValidAlertStop 或 xlValidAlertWarning
Operator	数据有效性验证运算符。可为以下常量之一：xlBetween、xlEqual、xlGreater、xlGreaterEqual、xlLess、xlLessEqual、xlNotBetween 或 xlNotEqual
Formula1	数据有效性验证等式中的第一表达式
Formula2	当Operator为xlBetween或xlNotBetween 时，需要使用的数据有效性验证等式的第二表达式（其他情况下，此参数被忽略）

其中第一参数 Type 表示有效性的验证类型，可选值及其含义如表 7-8 所示。

表 7-8　有效性验证的类型说明

名　　称	值	说　　明
xlValidateCustom	7	使用任意公式验证数据的有效性
xlValidateDate	4	日期值
xlValidateDecimal	2	数值
xlValidateInputOnly	0	仅在用户更改值时进行验证
xlValidateList	3	值必须存在于指定列表中
xlValidateTextLength	6	文本长度
xlValidateTime	5	时间值
xlValidateWholeNumber	1	全部数值

在本例中创建有效性验证时使用了 Type 和 Formula1 参数，表示有效性的条件是验证唯一性的公式。

（2）有效性验证的公式“=countif(" & Rng.Address & "," & Rng.Cells(1).Address(0, 0) & ")=1”用于判断区域 Rng 的数据的唯一性。假设 Rng 的地址是 B1:B51，那么此段代码产生的公式为“=Countif($B$1:$B$51,B1)=1”，其中 Countif 函数的第一参数是绝对引用，第二参数采用相对引用。

（3）Range.Address 表示单元格或者区域地址，其语法如下：

```
Range.Address(RowAbsolute, ColumnAbsolute, ReferenceStyle, External, RelativeTo)
```

其中前两个参数分别表示行或者列采用相对地址还是绝对地址，当赋值为 True 或者忽略参数时表示绝对地址，否则是相对地址；第三参数表示采用 R1C1 样式还是 A1 样式，默认为 A1 样式。

（4）对区域设置有效性的目的是验证区域中每个单元格是否具有唯一性，而最终产生红圈的代码是“ActiveSheet.CircleInvalid”，它表示将工作表中的无效数据项用红圈标示出来。不过它产生的红圈只能作为纠错用，即通过此方式发现哪个单元格有误，马上修改，这些红圈在保存工作簿时会自动消失。

本例案例文件请参考：..\ 第 7 章\ 7-11 标示选区中的重复值.xlsm

7.2　图形对象

图形对象包含图片、剪贴画、艺术字、形状、图表、文本框、SmartArt 和批注等内容，本节主要针对图片和批注进行案例演示，读者可以举一反三。

7.2.1　批量导入图片与图片名称

【案例要求】批量导入一个文件夹中的所有 jpg 图片和图片名称。

【知识要点】Shape.Select、Shape.Left、Shape.Top、Shape.Height、Shape.Width、shape.Placement 和 shape.ShapeRange.LockAspectRatio。

【程序代码】

```
Sub 批量导入图片()                                          '放置位置：模块中
  Dim str As String, n As Long, Paths, folder As FileDialog  '定义变量
  With Application.FileDialog(msoFileDialogFolderPicker)     '产生一个浏览窗口
    .AllowMultiSelect = False                                '不允许多选
    '如果未单击“取消”按钮则记录文件夹路径
```

```
        If .Show = True Then Paths = .SelectedItems(1)
    End With
    Paths = Paths & IIf(Right(Paths, 1) = "\", "", "\")
    Application.ScreenUpdating = False                    '关闭屏幕更新，提升速度
  '查找第一个符合条件的 jpg 格式的图片文件,提取文件名称
    str = Dir(Paths & "*.jpg")
    Do While Len(str) > 0                          '如果 jpg 格式的图片文件存在,就继续执行命令
        ActiveCell.Offset(n, 0) = Left(str, Len(str) - 4)    '将文件名称存放在单元格中（删除后缀名.jpg）
        '在当前表中插入图片,路径由 Paths 决定，文件的后缀名由 str 决定。插入的图片处于选中状态
        ActiveSheet.Shapes.AddPicture(Paths & str, False, True, ActiveCell.Offset(n, 1).Left, ActiveCell.Offset(n,
1).Top, ActiveCell.Offset(n, 1).Width, ActiveCell.Offset(n, 1).Height).Select
        Selection.Placement = xlMoveAndSize          '让图片的位置与大小随单元格的变化而变化
        n = n + 1                                    '记录插入的图片的个数
        str = Dir()                                  '查找下一个
    Loop
    Application.ScreenUpdating = True                '恢复屏幕更新
  '如果大于 0 个则提示插入的图片数量
    If i > 0 Then MsgBox "已插入" & n & "个图片！ ", vbOKOnly, "提示"
  End Sub
```

在执行过程时，程序会弹出一个“浏览”对话框供用户选择图片存放路径，如图 7-22 所示。

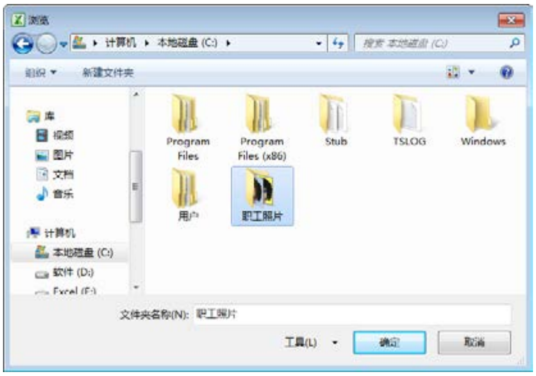


图 7–22　选择图片路径

当选择“职工照片”文件夹后单击“确定”按钮，程序会将该文件夹中所有图片及图片名称导入到活动单元格所在的区域，同时将图片插入到文件名称右边的单元格中，且将图片的上边距、左边距、高度和宽度调整为与单元格一致，效果如图 7-23 所示。

由于插入的图片高度和宽度由单元格决定，所以刚插入时图片会变形，此时按需要手动调整单元格的宽度和高度即可，调整行高与列宽后的效果如图 7-24 所示。



图 7–23　插入图片的效果

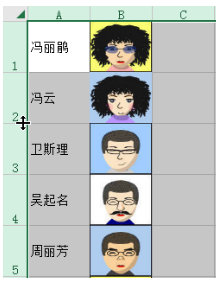


图 7–24　手动修改单元格高度与宽度后的效果

【语法补充】

（1）Application.FileDialog 表示产生一个对话框，让用户选择文件或者文件夹。当使用参数 msoFileDialogFolderPicker 时表示允许用户选择文件夹，而 SelectedItems(1)则表示被选择的文件夹的完整路径。

（2）FileDialog 产生的路径最后一位字符有可能是“\”也可能不是“\”，所以需要在代码中通过 IIf 函数判断，没有时则追加“\”。然后将它与变量 Str 所代表的文件名称组合成完整的包含路径的图片文件名称，并配合 ActiveSheet.Shapes.AddPicture 方法将该图片文件插入到工作表中。

（3）ActiveSheet.Shapes.AddPicture 方法的功能是将指定路径的图片文件插入到工作表中，同时指定它的大小、位置，具体语法如表 7-9 所示。

表 7-9　ActiveShet Shapes.AddPicture的参数说明

参　数	是否可选	类　型	说　明
FileName	必选	String	图片的路径和文件名
LinkToFile	可选	Variant	如果该参数值为True，则将图片链接到创建它的文件。如果该参数值为False，则将图片作为该文件的独立副本。默认值为False
SaveWithDocument	可选	Variant	如果该参数值为True，则将链接的图片与文档一起保存。默认值为False
Left	可选	Variant	图片的左边缘相对于绘图画布的位置，以磅为单位
Top	可选	Variant	图片的上边缘相对于绘图画布的位置，以磅为单位
Width	可选	Variant	图片的宽度，以磅为单位
Height	可选	Variant	图片的高度，以磅为单位

最后 4 个参数分别控制插入到工作表中的图片的左边距、上边距、宽度和高度。在本例中分别赋值为 ActiveCell.Offset(n, 1).Left、 ActiveCell.Offset(n, 1).Top、 ActiveCell.Offset(n, 1).Width 和 ActiveCell.Offset(n, 1).RowHeight，表示让图片刚好放在活动单元格下移 n 行、右移 1 列的单元格中。由于变量 n 在循环语句中会逐个累加，因此插入到工作表中的图片也会基于活动单元格向下逐一排放。

（4）如果通过录制宏生成插入图片的代码，那么宏代码会用 ActiveSheet.Pictures.Insert 方法来插入图片。而 ActiveSheet.Pictures.Insert 有一个明显的缺点：通过它插入的图片只有本机才能查看，把文件发给他人后，他人无法看到图片，因此本例改用 ActiveSheet.Shapes.AddPicture 方法来插入图片。

（5）Shape.ShapeRange.LockAspectRatio 属性用于控制图片的宽度与高度比例是否锁定，如果赋值为 msoTrue，则表示锁定比例，修改宽度时则高度相应变化，修改高度时宽度也相应变化。本例需要取消锁定，所以赋值为 msoFalse，使图片的宽度与高度可以随意修改。

（6）shape.Placement 属性表示对象附加到对象下方的单元格的方式，其包括三个可选项：xlFreeFloating（对象自由浮动）、xlMove（对象随单元格移动）、xlMoveAndSize（对象随单元格移动和调整大小），本例采用了第三种方式。

（7）Dir 函数用于获取文件名或者文件夹名，由于支持通配符，所以通常使用它在指定路径中查找文件。本例使用 Dir 函数获取 jpg 格式的图片文件名称，然后将它与路径组合作为 ActiveSheet.Shapes.AddPicture 的参数即可将图片插入到工作表中。配合循环语句则可以实现批量地插入图片。

（8）Dir 获取的文件名称总是包含后缀名，例如“.jpg”、“.doc”等。去除后缀名的办法有很多，本例采用的办法是从文件名称中提取总长度减 4 位的字符，原因是图片文件的后缀名称没有

3 位或者 5 位的。

本例案例文件请参考：..\ 第 7 章\ 7-12 批量导入图片到单元格.xlsm

7.2.2　统一表中所有图片大小及对齐图片

【案例要求】工作表中有诸多图片，由于是手动插入的，高度不统一，边距也不统一。现要求对它们统一高度，且调整所有图片的边距使其适应所在单元格的边距。

【知识要点】DrawingObjects.ShapeRange.LockAspectRatio、DrawingObjects.ShapeRange.Placement、Shape.TopLeftCell 和 Shape.Type。

【程序代码】

```
Sub 统一图片的大小及对齐图片()                        '放置位置：模块中
   Application.ScreenUpdating = False                 '关闭屏幕更新
    With ActiveSheet.DrawingObjects                   '引用所有图形对象
    If .Count = 0 Then Exit Sub                       '如果没有图片则退出程序
        .ShapeRange.LockAspectRatio = msoTrue         '锁定图片的长宽比例
        .Placement = xlMove                           '位置随单元格移动，但大小固定
    End With
    Dim sh As Shape, ShapeHeight As Integer           '声明变量
    ShapeHeight = 80                                  '对表示高度的变量赋值 80，可以随时修改
    For Each sh In ActiveSheet.Shapes                 '遍历工作表中的所有图形对象
        If sh.Type = msoPicture Then                  '如果是图片
          '将图片的左边距设置为图片左上角的单元格的左边距加 1
           sh.Left = sh.TopLeftCell.Left + 1
          '将图片的上边距设置为图片左上角的单元格的上边距加 1
           sh.Top = sh.TopLeftCell.Top + 1
          '将图片的高度修改为指定的高度减 2（避免盖住边框线）
           sh.Height = ShapeHeight – 2
          '将单元格高度设置为指定的高度，从而刚好容纳图片
           sh.TopLeftCell.RowHeight = ShapeHeight
        End If
Next sh
Application.ScreenUpdating = True                     '恢复屏幕更新
End Sub
```

假设工作表中的图片像图 7-25 一样凌乱摆放，执行过程“统一图片的大小及对齐图片”后可以将它们修改为如图 7-26 所示效果，每个图片与它所在单元格左对齐、上对齐，同时将单元格的高度也一并统一，使其能刚好容纳图片又不至于让图片盖住单元格边框。



图 7–25　凌乱的产品展示



图 7-26　调整图片大小及对齐后的效果

【语法补充】

（1）图片的 LockAspectRatio 属性和 Placement 属性都可以一次性修改，所以本例先统一修改图片的这两个属性，然后再通过循环语句修改图片的边距和大小。此方式可以提升代码执行速度。

（2）由于在设置图片大小时也需要调整单元格的高度，而修改单元格的高度时可能会造成图片错位，所以在循环之前需要将图片的 Placement 属性修改为 xlMove，从而确保图片与原来的单元格相对位置不变。

（3）本例中变量 ShapeHeight 表示单元格的高度，可以随意修改，但由于单元格的高度有限制（0~409），因此对 ShapeHeight 赋值时实际需要采用小于等于 409 且大于 0 的值。

在本例中，将图片的高度设置为 ShapeHeigh 减 2，目的是使图片上边距和下边距都比单元格的高度少 1，从而避免盖住单元格的边框。

（4）Shape.Type 表示图形对象的类型，本例只修改图片的高度与边距，所以通过 If 语句判断其类型名称，如果是图表、文本框、艺术字、剪贴画等对象则自动忽略。

（5）Shape.TopLeftCell 表示图形对象左上角的单元格，Shape.BottomRightCell 则表示右下角的单元格。

本例案例文件请参考：..\ 第 7 章\ 7-13 统一表中所有图片大小及左对齐.xlsm

7.2.3　插入图片到选区中

【案例要求】将任意选中的图片插入到选区中，图片的大小刚好等于选区的大小。

【知识要点】TypeName、Range. Areas、Application.GetOpenFilename 和 Shapes.AddPicture。

【程序代码】

```
Sub 插入图片到选区中()                              '放置位置：模块中
    If TypeName(Selection) = "Range" Then          '如果当前选择的对象是单元格
    '如果选区的区域数量大于 1，那么提示只能选择一个区域，然后结束过程
        If Selection.Areas.Count > 1 Then MsgBox "只能选择一个区域", vbOKOnly, "友情提示": Exit Sub
        Dim PicFile As String                      '声明一个变量，用于储存图片路径
        PicFile = Application.GetOpenFilename("图片文件 (*.jpg;*.png;*.bmp), *.jpg;*.png;*.bmp", , "请选择图片文件,只能选择单个", , False)
```

```
        If PicFile = "False" Then Exit Sub            '如果用户单击了“取消”按钮。那么结束过程
        '将变量 PicFile 代表的图片插入到工作表中，其边距大小由选区决定
        ActiveSheet.Shapes.AddPicture  PicFile,  False,  True,  Selection.Left,  Selection.Top,  Selection.Width,
Selection.Height
      Else '否则（即选择了图形对象）
        MsgBox "请选择单元格区域。", vbOKOnly, "友情提示" '提示用户
      End If
    End Sub
```

在选择 B2:D10 区域后执行过程“插入图片到选区中”，程序会弹出如图 7-27 所示的图片浏览对话框，在对话框的左上角有提示文字“请选择图片文件,只能选择单个”。



图 7–27　图片浏览对话框

任意选择一个图片并单击“打开”按钮，程序会将选中的图片插入到 B2:D10 区域，且图片的大小刚好等于 B2:D10 区域的大小，效果如图 7-28 所示。

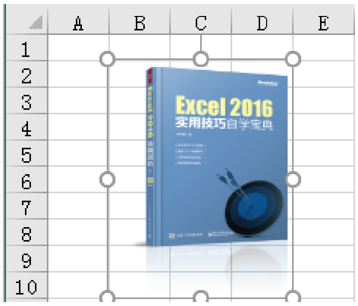


图 7–28　在 B2:D10 区域中插入图片

【语法补充】

（1）Selection 代表当前选中的对象，它有可能是图片、图表或者单元格，可以通过 TypeName 函数来判断。当对象是单元格时返回值为 Range，当对象是图片时返回值为 Picture，当对象是图表时返回值为 ChartArea。

（2）使用等号判断两个字符串是否相等时，首字母需要区分大小写，因此代码“TypeName(Selection) = "Range"”不能写作“TypeName(Selection) = "range"”。

（3）Range.Areas.Count 代表 Range 对象的区域数量，当按住 Ctrl 键多选区域时，Range.Areas.Count 的值会大于 1。

（4）Application.GetOpenFilename 表示弹出一个标准的“打开”对话框，并获取用户选中

的文件的名称。当执行多选操作时返回值是数组，当执行选择单元格操作时返回值则是单个文件的完整路径名称。Application.GetOpenFilename 的具体语法如下：

```
Application.GetOpenFilename(FileFilter, FilterIndex, Title, ButtonText, MultiSelect)
```

该方法的具体语法如表 7-10 所示。

表 7-10　Application.GetOpenFilename的参数说明

名　称	必选/可选	说　明
FileFilter	可选	一个指定文件筛选条件的字符串
FilterIndex	可选	指定默认文件筛选条件的索引号，取值范围为1到由 FileFilter 所指定的筛选条件数目。如果省略该参数，或者该参数的值大于可用筛选条件数，则使用第一个文件筛选条件
Title	可选	指定对话框的标题。如果省略该参数，则标题为“打开”
ButtonText	可选	仅限 MaCintosh，Windows中使用时留空即可
MultiSelect	可选	如果赋值为True则允许选择多个文件；如果赋值为False，则只允许选择一个文件名。默认值为False

其中 FileFilter 参数最重要，它控制对话框中可以看到的文件的类型，即后缀名，允许同时指定多个文件类型。本例指定了 3 种常见的图片类型名称，它们之间用分号隔开。

第 5 参数 MultiSelect 赋值为 True 表示可以选择多个文件，它的返回值为数组。本例中只需要插入单个图片，因此将它赋值为 False，返回值为图片的路径，String 型。

（5）对于本例还可以加一些限制，避免图片过多，例如不允许在选择整行、整列时插入图片。读者可自行修改代码完成。

本例案例文件请参考：..\ 第 7 章\ 7-14 将图片插到选区中.xlsm

7.2.4　插入带图片背景的批注

【案例要求】在活动单元格插入批注，将指定的图片填充到批注的背景中，允许随意调整图片宽度与高度的比例。

【知识要点】Range.ClearComments、Range.AddComment、Shape.Fill、Shape.LockAspectRatio、Shape.Width 和 Shape.Height。

【程序代码】

```
Sub 插入图片标注()                                  '放置位置：模块中
    Dim 图片名称 As String, 规格 As String          '声明变量，前者表示图片路径，后者表示图片规格
   '让用户指定图片的路径
    图片名称 = Application.GetOpenFilename( "图片(*.bmp;*.gif;*.tif;*.jpg;*.jpeg),*.bmp;*.gif;*.tif;*.jpg;*.jpeg")
    If 图片名称 = "False" Then End                   '如果单击“取消”按钮则终止程序
Start:    '设置一个标签
    Do   '循环执行命令
        '让用户指定图片显示在批注中的规格：宽度与高度的比例
        规格 = Application.InputBox("请指定批注的宽度与高度比例，例如[1:2]表示高度是宽度的两倍", "比例", "1:1.5", , , , , 2)
        If 规格 = "False" Then End                   '如果单击“取消”按钮则终止程序
        '如果输入了冒号就结束循环，否则提示用户，然后继续循环命令
        '用户可能输入半角或者全角的冒号，所以为了统一，采用 Replace 将全角的冒号转换成
        '半角的冒号，此时不管如何输入都将当作半角冒号处理
         If InStr(Replace(规格, "：", ":"), ":") Then Exit Do Else MsgBox "请输入冒号！", vbOKOnly, "友情提示"
```

```
        Loop
        Dim LeftVal As String, RightVal As String
        LeftVal = Left(规格, InStr(规格, ":") - 1)              '提取冒号左边的字符
        RightVal = Right(规格, Len(规格) - InStr(规格, ":"))    '提取冒号右边的字符
        '如果比例中代码宽与高的字符都是数字
        If IsNumeric(LeftVal) And VBA.IsNumeric(RightVal) Then
            '如果有任意一个字符等于 0 就返回标签 Start 处，等待重新输入
            If LeftVal <= 0 Or RightVal <= 0 Then MsgBox "只能输入大于 0 的正数", vbOKOnly, "友情提示": GoTo Start
            '如果比例小于等于 0.1 或者大于等于 10 也返回标签处，等待重新输入
            If LeftVal / RightVal <= 0.1 Or LeftVal / RightVal >= 10 Then MsgBox "宽与高不适相差 10 倍以上", vbOKOnly, "友情提示": GoTo Start
            ActiveCell.ClearComments                   '清除活动单元格的批注（假设有的话）
            With ActiveCell.AddComment                 '添加新的批注
                .Shape.Fill.UserPicture 图片名称         '将图片设置为批注的背景
                .Shape.LockAspectRatio = msoFalse      '取消“锁定纵横比”
                .Shape.Width = 100                     '将批注的宽度设为 100，可以随意修改该值
                '将批注的宽度设置为 100 乘以高与宽的比例
                .Shape.Height = 100 * RightVal / LeftVal
                .Visible = False                       '隐藏批注
            End With
        Else
            MsgBox "只能是数字"                          '如果不是数字则提示
            GoTo Start                                 '返回标签 Start 处继续执行循环语句
        End If
    End Sub
```

当执行过程“插入图片批注”后会弹出选择图片文件的对话框，支持 bmp、gif、tif、jpg 和 jpeg 格式，如图 7-29 所示。

当指定图片路径后程序会弹出第二个对话框，供用户指定图片的宽度与高度的比例。如图 7-30 所示。如果用户输入的字符串中没有冒号，那么程序会提示用户输入冒号，然后继续弹出输入框等待用户输入图片的宽度与高度的比例；如果用户输入的字符串中除冒号外有其他文本字符，那么同样在提示用户后继续弹出输入框等待用户输入图片的宽度与高度的比例。只有在输入正确的数据后才会执行下一步。

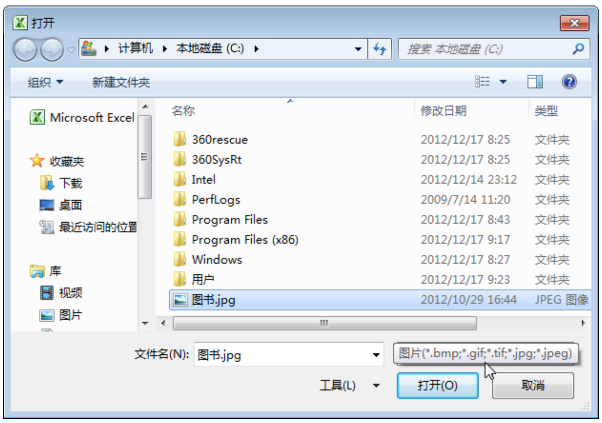


图 7–29　选择图片文件

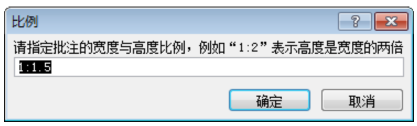


图 7–30　在输入框中指定批注的规格

如果图片的比例输入了“1:0.8”，那么图片批注效果如图 7-31 所示；如果图片的比例输入了

"1:1.3"，那么图片批注效果如图 7-32 所示：

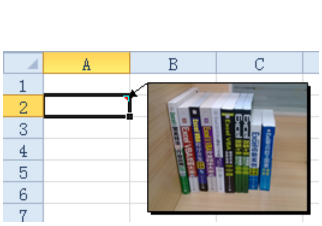


图 7–31　比例为 1 : 0.8 时的批注

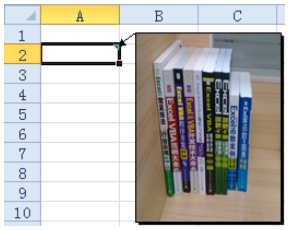


图 7–32　比例为 1 : 1.3 时的批注

【语法补充】

（1）当 Application.GetOpenFilename 方法的第 5 参数使用 False 时，它的返回值是 String 型，所以当用户选择[取消]按钮时，Application.GetOpenFilename 方法的返回值是"False"，此处要注意首字母大写。

（2）在输入图片规格时，由于用户输入的冒号可能是全角也可能是半角的，因此代码中采用了自动替换的方式，从而确保 Instr 函数总能正确地计算冒号的位置。

（3）为了避免用户在冒号左边或者右边输入文本，本例采用 IsNumeric 函数执行判断，当输入的是文本时则执行 Start 标签后的代码，等待用户输入新的比例。如果输入的字符串符合条件则执行插入批注、填充图片等操作。本例中 Do Loop 循环和标签 Start 相当重要。

（4）Shape.Fill 表示对图形对象进行填充，有多种填充方式。当使用"UserPicture"属性时表示使用图片填充，参数为含有完整路径的图片名称。

（5）图形对象默认都是锁定纵横比的，所以需要在修改宽度或者高度前必须将其 LockAspectRatio 属性赋值为 msoFalse。如果要求图片不能变形，则采用默认值即可。

本例案例文件请参考：..\ 第 7 章\ 7-15 创建带图片背景的批注.xlsm

7.3　工作表对象

在 VBA 中，WorkSheet 表示工作表对象，而 Worksheets 表示工作表集合。在实际工作中，工作表对象以及工作表对象的属性、方法和事件都应用较频繁，本节提供工作表对象的属性、方法方面的案例演示，而工作表事件则将在第 8 章中详细阐述。

7.3.1　显示所有隐藏的工作表

【案例要求】工作簿中存在隐藏的工作表，要求一键显示所有工作表。

【知识要点】WorkSheet.Visible。

【程序代码】

```
Sub 显示所有工作表()                              '放置位置：模块中
    Dim sht As Worksheet                         '声明变量
    If ActiveWorkbook.ProtectStructure Then      '如果工作簿的结构受保护
        End                                      '结束过程
    Else                                         '否则
        For Each sht In Sheets                   '遍历所有表
            sht.Visible = xlSheetVisible         '显示工作表
        Next sht
```

```
        End If
    End Sub
```

执行以上过程后，如果工作簿的结构被保护则自动终止程序，因为在保护工作簿的结构的前提下是不可能显示隐藏工作表的。

如果工作簿为未保护结构，那么工作簿中的隐藏工作表将自动显示出来。

【语法补充】

（1）工作表的 Visible 属性有三种状态，包含 xlSheetVisible、xlSheetHidden 和 xlSheetVeryHidden，其中 xlSheetVisible 表示显示状态，xlSheetHidden 表示隐藏状态，xlSheetVeryHidden 表示深度隐藏状态。在 xlSheetHidden 状态下，可以选择工作表标签的快捷菜单中的“取消隐藏”命令实现显示隐藏的工作表，但是当工作表处于深度隐藏的状态时，工作表标签的快捷菜单中的“取消隐藏”命令将处于灰度禁用状态，需要用 VBA 代码才可以显示被隐藏的工作表。

（2）当要保护工作簿时可以选择是保护窗口还是保护结构，如果保护了工作簿的结构，就不可以隐藏该工作簿中的任意工作表，也不可以取消工作表的隐藏属性。所以为了避免代码出错，有必要检查工作簿的结构是否处于保护状态。

（3）由于显示工作表不会改变工作表的顺序和位置，所以可以使用 For Each...Next 循环，若是在循环体中按某条件删除工作表，那么需要使用 For Next 语句从后向前循环，其步长值为-1。

本例案例文件请参考：..\ 第 7 章\ 7-16 显示所有隐藏工作表.xlsm

7.3.2　创建以本月每日日期命名的工作表

【案例要求】批量创建工作表，数量等于本月的总天数，每个工作表以本月每一天的日期命名。

【知识要点】Worksheets.Add 和 Sheets.Count。

【程序代码】

```
    Sub 批量创建以本月每日日期命名的工作表()                         '放置位置：模块中
       '当有错误时继续执行下一步（存在同名工作表时会出错）
        On Error Resume Next
        Dim Days As Byte, Item As Byte                              '声明变量
       '计算本月有多少天（下个月 0 日即本月最后一天）
        Days = Day(DateSerial(Year(Date), Month(Date) + 1, 0))
        For Item = 1 To Days                                        '从 1 到本月最后一天
        '在最后一个工作表的后面新建工作表，并对其命名为升序的日期序列值
            Worksheets.Add(, Sheets(Sheets.Count), 1, xlWorksheet).Name = Year(Date) & "-" & Month(Date) &
"-" & Item
        Next Item
    End Sub
```

当执行过程“批量创建以本月每日日期命名的工作表”后，假设执行代码的时间为 2016 年 12 月，那么程序会在工作簿的最后一个工作表右边逐一创建 12 月 1 日到 31 日命名的 31 个工作表，如图 7-33 所示。

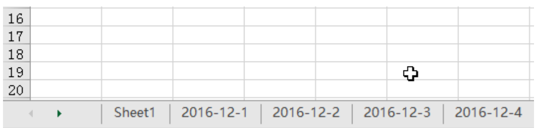


图 7-33　批量创建本月日期命名的工作表

【语法补充】

（1）Worksheets.Add 表示创建工作表，语法如下：

```
Worksheets.Add(Before, After, Count, Type)
```

其中四个参数都是可选参数，四个参数全忽略时表示在活动工作表的前面插入一个工作表。本例中“Worksheets.Add(, Sheets(Sheets.Count), 1, xlWorksheet)”表示在最右方插入一个工作表。

Sheets.Count 表示表的总数量，而 Sheets(Sheets.Count)则代表最后一个表。本例使用 Sheets(Sheets.Count)作为 Worksheets.Add 的第二参数，表示新建的表放在最后一个表的右方。

当然本例也可以一次性创建所有工作表，然后逐一命名，代码如下：

```
Sub 批量创建以本月每日日期命名的工作表()                    '放置位置：模块中
    '当有错误时继续执行下一步（存在同名工作表时会出错）
    On Error Resume Next
    Dim Days As Byte, Item As Byte, ShtCount As Integer        '声明变量
   '计算本月有多少天（下个月 0 日即本月最后一天）
    Days = Day(DateSerial(Year(Date), Month(Date) + 1, 0))
    ShtCount = Sheets.Count                                    '统计当前的工作表数量
    Worksheets.Add , Sheets(Sheets.Count), Days, xlWorksheet   '批量创建工作表
    For Item = ShtCount + 1 To Sheets.Count                    '遍历所有新建的工作表
      '对新建的工作表命名，“Item - ShtCount”表示新表的序号减去原来的工作表总数量
       Sheets(Item).Name = Year(Date) & "-" & Month(Date) & "-" & Item - ShtCount
    Next Item
End Sub
```

代码中的 Date 代表今天的日期，而 Year(Date)代表今年的年份，Month(Date)代表本月的月份。

（2）DateSerial 函数可以产生一个包含年月日的日期值，它的三个参数分别代表年、月、日。例如 DateSerial(2017,5,9)返回代表 2017 年 5 月 9 日的日期值。

DateSerial 函数具有智能调节功能，当月和日超过正常范围时可以自动调整为有效值。例如 2012 年 13 月会调整为 2013 年 1 月，而下个月 0 日则会自动调整为本月最后一日。本例正是利用这个原理，先产生下个月 0 日的日期值，当 DateSerial 函数将它转换成本月最后一天的日期值后再用 Day 函数计算本月总共有多少天。

在代码中 Day(DateSerial(Year(Date), Month(Date) + 1, 0))表示本月的总天数，要计算上个月的总天数则改用 Day(DateSerial(Year(Date), Month(Date) , 0))。

本例案例文件请参考：..\ 第 7 章\ 7-17 批量创建工作表.xlsm

7.3.3　保护所有公式

【案例要求】保护活动工作簿中所有工作表的所有公式，避免误删公式。但同时不能影响其他单元格的正常操作。

【知识要点】Worksheets.Count、Worksheet.UsedRange、Worksheet.Protect 和 Worksheet.ProtectContents。

【程序代码】

```
Sub 保护公式()    '对活动工作簿中所有未保护的工作表生效（放置位置：模块中）
    Dim Item As Integer                                '声明变量
    On Error Resume Next                               '有错误时继续执行下一步
```

```
    For i = 1 To Worksheets.Count                          '遍历所有工作表（不能使用 Sheets）
      If Not Worksheets(i).ProtectContents Then            '如果工作表未保护
        Worksheets(i).UsedRange.Locked = False             '解除已用数据区域的锁定属性
        Worksheets(i).UsedRange.FormulaHidden = False      '解除已用数据区域的隐藏公式属性
         With Worksheets(i).UsedRange.SpecialCells(xlCellTypeFormulas, 23) '引用公式区域
          .Locked = True                                   '锁定单元格
          .FormulaHidden = True                            '隐藏公式
          '保护工作表，密码为空文本，允许一切操作，避免保护公式后影响其他操作
          Worksheets(i).Protect Password:="", DrawingObjects:= False, _
          Contents:=True, Scenarios:= False, AllowFormattingCells:=True, _
          AllowFormattingColumns:=True, AllowFormattingRows:=True, _
          AllowInsertingColumns:=True, AllowInsertingRows:=True, _
          AllowInsertingHyperlinks:=True, AllowDeletingColumns:=True, _
          AllowDeletingRows:=True, AllowSorting:=True, AllowFiltering:=True, _
          AllowUsingPivotTables:=True
        End With
       End If
    Next i
End Sub
```

假设工作表中有如图 7-34 所示的数据，表中包含公式和非公式，执行过程“保护公式”后所有工作表中的公式都会处于隐藏状态，在选择公式所在单元格时无法在编辑栏中看到公式本身，也无法删除公式，如图 7-35 所示。

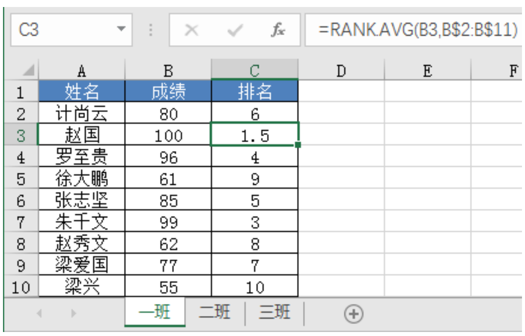


图 7–34　成绩表

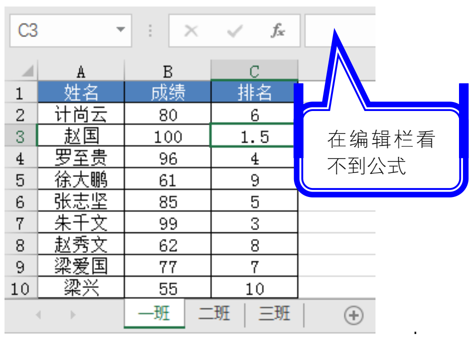


图 7–35　保护公式后的状态

【语法补充】

（1）保护公式是针对工作表的，所以循环语句中使用 Worksheets 而不用 Sheets。

（2）Worksheet.ProtectContents 表示工作表的保护属性，值为 True 时表示已保护，值为 False 时表示未保护。

（3）Range.Locked 和 Range.FormulaHidden 两个属性分别对应“设置单元格格式”对话框中“保护”选项卡的“锁定”与“隐藏”两个属性。本例的思路是将已用区域中所有单元格的“锁定”与“隐藏”两个属性赋值为 False，然后对 SpecialCells 获得的公式区域的两个属性赋值为 True，从而将公式与非公式区别开来。

（4） Worksheet.Protect 表示保护工作表，其语法如下：

```
    Worksheet.Protect(Password, DrawingObjects, Contents, Scenarios, UserInterfaceOnly, AllowFormattingCells,
AllowFormattingColumns, AllowFormattingRows, AllowInsertingColumns, AllowInsertingRows,
AllowInsertingHyperlinks, AllowDeletingColumns, AllowDeletingRows, AllowSorting, AllowFiltering,
AllowUsingPivotTables)
```

其中 Password 参数表示密码，本例中采用了空文本，表示只保护工作表但不设密码。其他

的各参数对应如图 7-36 所示的复选框。由于参数较多，请读者参阅 VBA 帮助中的说明，查询帮助的关键字为 “Worksheet.Protect”。

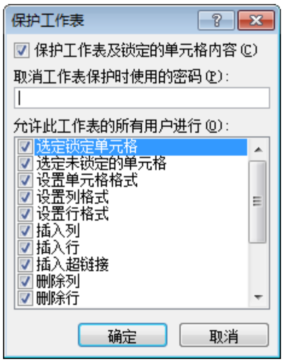


图 7–36　保护工作表对话框

（5）本例使用了空密码保护工作表，读者可以根据喜好使用其他密码。不过如果工作表已经处于保护状态，那么本例的代码会自动忽略此工作表，只有手动解密后才能保护公式。

本例案例文件请参考：..\ 第 7 章\ 7-18 保护所有公式.xlsm

7.3.4　批量重命名表

【案例要求】利用选区的数据对表批量重命名，选区中的单元格顺序与表顺序一致。

【知识要点】Sheet.Name。

【程序代码】

```
Sub 批量重命名()                                          '放置位置：模块中
    On Error Resume Next                                  '有错误时继续执行
    Dim Rng As Range, Item As Integer                     '声明变量
    If TypeName(Selection) <> "Range" Then Exit Sub       '如果选择的对象不是单元格就结束过程
                                                          '如果选区中有空白单元格则结束过程
    If WorksheetFunction.CountBlank(Selection) > 0 Then MsgBox "选区中不能有空白单元格": Exit Sub
                                                          '如果选区中不存在重复值
    If Evaluate("=Sum(1/CountIf(" & Selection.Address & "," & Selection.Address & "))") = Selection.Count
Then
        For Each Rng In Selection                         '遍历选区所有单元格
            Item = Item + 1                               '累加变量
            Sheets(Item).Name = Rng.Value                 '对工作表命名
        Next Rng
    End If
End Sub
```

对于如图 7-37 所示的这种工作簿，当选择 A1:A7 后再执行过程 “批量重命名” 可以得到如图 7-38 所示结果。

不过，如果选区中有空白单元格或者选区中存在重复值，那么过程不会执行，因为不可以用空值对表进行命名，也不可以对多个表使用相同的名称进行命名。



图 7–37　待重命名的工作表



图 7–38　批量命名效果

【语法补充】

（1）Excel VBA 无法直接判断选区中是否存在空白单元格，不过工作表函数 CountBlank 能计算区域中的空白单元格个数，所以本例借助它计算空白单元格的数量，如果大于 0 则表示选区中存在空白单元格。

（2）Excel VBA 也无法直接判断一个区域中是否存在重复值，通常采用循环语句配合集合、字典等技术来实现。也可以通过 Excel 的数组公式实现，所以本例首先产生一个数组公式"=Sum(1/CountIf(" & Selection.Address & "," & Selection.Address & "))"，此公式用于计算选区中的不重复数据个数，然后通过 Evaluate 函数计算该公式的值，如果该值等于选区的单元格个数，则表示不存在重复值。

由于本例中 Selection 代表 A1:A7 区域，所以以上代码产生的公式如下：

```
=Sum(1/CountIf($A$1:$A$7, $A$1:$A$7))
```

（3）Application.Evaluate 方法用于将文本表达式转换为一个对象或者一个值，转换结果是对象引用还是值，则取决于该文本表达式的含义，示例如下。

MsgBox　Evaluate("1+2")——转换结果为 3，它将"1+2"识别为公式。

MsgBox　Evaluate("Sheet1!a1").Address——转换结果为$A$1，它将"Sheet1!a1"识别为对象，即 Range 对象，代表 Sheet1 工作中的 A1 单元格。

MsgBox Evaluate("Sum（A1:A10）+sqrt(100)") ——转换结果为 A1:A10 区域的值加 10 的计算结果，具体值为多少取决于 A1:A10 区域的值。

Application.Evaluate 方法通常用于计算文本形式的公式，将其转换成计算结果。不过它有长度限制，参数不能超过 256 位。

（4）由于有些字符不适合用于工作表名称，例如星号（*）、除号（/）、半角状态的大于号（>）等，所以如果单元格中存在这些字符中的任意一个时就会命名不成功，从而造成代码中断，使后面代码也无法执行，所以有必要在代码的前面加入防错语句"On Error Resume Next"。在本书的第 9 章会详细讲解防错语句的用法。

本例案例文件请参考：..\ 第 7 章\ 7-19 批量命名工作表.xlsm

7.3.5　查找所有工作表中有循环引用的单元格

【案例要求】如果工作簿中存在循环引用的公式，那么在打开工作簿时会弹出"循环引用警告"，然后在状态栏显示其中一个有循环引用的单元格的地址。假设工作簿中有多个单元格存在循环引用，那么只能使用 VBA 来获得这些单元格的地址。

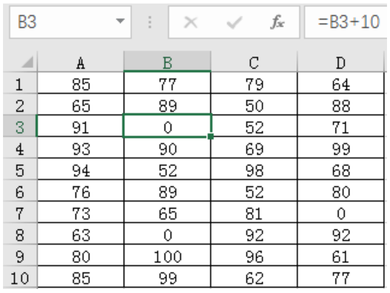
【知识要点】Worksheet.CircularReference。

【程序代码】

```
Sub 查找有循环引用的所有单元格()                    '放置位置：模块中
```

```
    Dim Rng As Range, Goal As Range                    '声明变量
    Do                                                 '开始循环
      Set Rng = Worksheets("Sheet1").CircularReference '引用第一个有循环引用的单元格
      If Rng Is Nothing Then                           '如果不存在有循环引用的单元格
        Exit Do                                        '退出循环
      Else                                             '否则
         '在单元格的公式前添加一个字符，使单元格不再有公式
          Rng = "@" & Rng.Formula
         '如果变量 Goal 未初始化，那么将变量 Rng 赋值给它，否则将 Goal 与 Rng 合并
          If Goal Is Nothing Then Set Goal = Rng Else Set Goal = Union(Goal, Rng)
      End If
    Loop
    If Not Goal Is Nothing Then                        '如果工作表中存在循环引用（Goal 已初始化）
      MsgBox "有循环引用的单元格包括：" & Goal.Address '报告所有循环引用的单元格的地址
      For Each Rng In Goal                             '遍历所有循环引用的单元格
          Rng = Mid(Rng.Formula, 2, 9999)              '删除添加的字符，使其还原为公式
      Next Rng
    End If
End Sub
```

假设工作表中有如图 7-39 所示数据，其中 B3、D7 和 B8 单元格有循环引用的公式，那么执行过程“查找有循环引用的所有单元格”后将得到如图 7-40 所示的结果。


B3 =B3+10

	A	B	C	D
1	85	77	79	64
2	65	89	50	88
3	91	0	52	71
4	93	90	69	99
5	94	52	98	68
6	76	89	52	80
7	73	65	81	0
8	63	0	92	92
9	80	100	96	61
10	85	99	62	77

图 7–39　包含循环引用的工作表

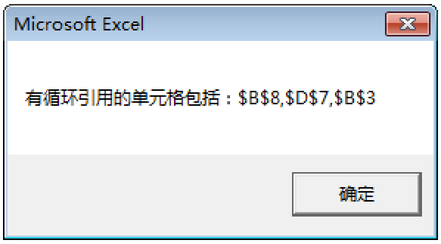


图 7–40　查找到的循环引用的单元格

【语法补充】

（1）Worksheet.CircularReference 用于返回工作表中第一个具有循环引用的单元格，如果不存在则返回 Nothing。

由于总是返回目标中的第一个，所以本例配合 Do Loop 循环语句逐一提取所有目标，最后使用 Union 方法将它们合并为一个对象。

（2）公式以等号开头，在等号前添加任意等号以外的字符，都可以让公式变成字符串。

（3）在利用 Mid 函数提取字符串时，如果第 3 参数的值超过了字符的总长度，那么返回值时会智能地判断长度，忽略多余的部分。所以本例的代码中 Mid 第 3 参数采用 9999，而非计算公式的实际长度，此方式可以缩减代码长度。

本例案例文件请参考：..\ 第 7 章\ 7-20 查找有循环引用的所有单元格.xlsm

7.3.6　对职工表按学历排序

【案例要求】对用户信息表按学历升序排序，学历的排序条件为：小学、初中、高中、大学。

【知识要点】WorkSheet.Sort。

【程序代码】

```
    Sub  自定义排序()                              '放置位置：模块中
      With ActiveSheet.Sort                      '引用 Sort 对象
        .SortFields.Clear                        '清除其他排序条件
         '对 B1 到 B 列最后一个非空单元格的区域添加自定义排序的字段,也就是说对本句代码指定排序
'条件及条件所在列，下一句代码指定参与排序的区域
        .SortFields.Add Key:=Range("B1:B" & Cells(Rows.Count, "B").End(xlUp).Row), CustomOrder:="小学,初
中,高中,大学"
       '设置参与排序的区域地址
        .SetRange Range("A1:B" & Cells(Rows.Count, "B").End(xlUp).Row)
        .Header = xlYes                          '表头不参与排序
        .Orientation = xlSortColumns             '按列排序
        .Apply                                   '执行排序
      End With
    End Sub
```

以图 7-41 的数据为例，执行过程“自定义排序”后，原来乱序的用户信息表将排列成如图 7-42 所示的样式。



	A	B	C
1	姓名	学历	
2	陈星望	小学	
3	穆容秋	初中	
4	张庆	小学	
5	赵秀文	小学	
6	李湖云	大学	
7	张大中	高中	
8	张珍华	小学	
9	张三月	初中	
10	刘子中	小学	
11	刘专洪	大学	

图 7–41　待排序的职工信息表



	A	B	C
1	姓名	学历	
2	陈星望	小学	
3	张庆	小学	
4	赵秀文	小学	
5	张珍华	小学	
6	刘子中	小学	
7	穆容秋	初中	
8	张三月	初中	
9	张大中	高中	
10	李湖云	大学	
11	刘专洪	大学	

图 7–42　自定义排序结果

【语法补充】

（1）WorkSheet.Sort 是从 Excel 2007 版本开始新增的一个工作表属性，同时 Sort 也是一个对象，可用它对工作表中的数据排序，而在 Excel 2003 中采用 Range.Sort 方法来排序，两者的操作方式大不相同。

使用 WorkSheet.Sort 对象执行排序时，WorkSheet.Sort 会保留上次的排序条件，所以通常在排序前有必要通过“SortFields.Clear”方法清除原有的条件，然后通过“SortFields.Add”方法添加新的排序条件。

（2）SortFields.Add 表示添加新的排序条件，其语法如下：

```
SortFields.Add(Key, SortOn, Order, CustomOrder, DataOption)
```

SortFields.Add 的参数说明表 7-11 所示。

表 7-11　SortFields.Add的参数列表与含义说明

名　　称	说　　明
Key	指定用于排序的键值，通常表示排序条件所在列，例如“D2:D10”
SortOn	排序的依据，例如是按值(xlSortOnValues)还是按单元格颜色（xlSortOnCellColor）排序的
Order	指定排序次序，赋值xlAscending时表示升序，赋值xlDescending时表示降序
CustomOrder	指定自定义排序次序的条件，例如“小学,初中,高中,大学”
DataOption	文本排序方式。赋值xlSortNormal，表示分别对数字和文本数据进行排序；赋值为xlSortTextAsNumbers，表示将文本作为数字型数据进行排序

SortFields.Add 的 5 个参数都是可选参数，本例中使用自定义序列排序，所以指定 Key 和 CustomOrder 参数即可。其中 Key 表示排序条件的所在区域，后者表示自定义的排序条件。CustomOrder 参数的条件是一个逗号分隔开的字符串，不能使用数组。

（3）SortFields.Add 方法的 Key 参数用于指定排序条件的区域，而 Sort. SetRange 方法则用于指定整个参与排序的区域。例如对 A1:G10 区域排序，那么用 Sort. SetRange 方法指定 A1:G10，如排序条件在 F 列，那么就用 SortFields.Add 方法的 Key 参数指定该区域。

（4）Sort.Header 表示是否让表头参与排序，为赋值为 xlYes 时，表头（首行）不参与排序。

（5）Sort.Orientation 表示纵向排序还是横向排序，本例赋值 xlSortColumns，表示纵向。

（6）Sort.Apply 表示执行排序，即前面的代码只是设置排序的方式，此句才是用于执行排序操作的语句。

本例案例文件请参考：..\ 第 7 章\ 7-21 按学历排序.xlsm

7.3.7 创建工作表目录

【案例要求】创建一个名为“工作表目录”的工作表，然后对其他所有工作表创建带链接的目录，保存在“工作表目录”中。

【知识要点】Sheet.Delete、Sheets.Add、WorkSheet.Hyperlinks.Add 和 Sheets.Count 。

【程序代码】

```
Sub 创建工作表目录()                                    '放置位置：模块中
    Application.ScreenUpdating = False                   '关闭屏幕更新
   '有错误时继续下一步（当没有“工作表目录”时删除工作表的代码会出错，采用本代码后可避免）
    On Error Resume Next
    Application.DisplayAlerts = False                    '关闭提示（删除非空工作表时会弹出提示框）
    Sheets("工作表目录").Delete                          '删除已有的工作表目录（假设有的话）
    Application.DisplayAlerts = True                     '恢复提示
   '在最前面创建一个新工作表，并命名为“工作表目录”
    Sheets.Add(before:=Sheets(1)).Name = "工作表目录"
   '写入标题，采用数组形式，可以一次性写入多个单元格
    Range("A1:B1") = [{"编号", "目录"}]
    For i = 2 To Sheets.Count                            '遍历工作表目录以外的所有工作表
        Cells(i, 1).Value = i - 1                        '在单元格中写入序号
        '在单元格中创建链接
        ActiveSheet.Hyperlinks.Add Anchor:=Cells(i, 2), Address:="", SubAddress:="'" & Sheets(i).Name &
"'" & "!A1", TextToDisplay:=Sheets(i).Name, ScreenTip:="单击打开：" & Sheets(i).Name
    Next
    Range("A2").Select                                   '选择 A2 单元格
    ActiveWindow.FreezePanes = True                      '冻结窗格（让首行固定，方便查看）
    Application.ScreenUpdating = True                    '恢复屏幕更新
End Sub
```

假设活动工作簿中有如图 7-43 所示的工作表，执行过程“创建工作表目录”后将得到如图 7-44 所示的结果，当光标停在指定目录时会有相应的提示，单击则能打开对应的工作表。

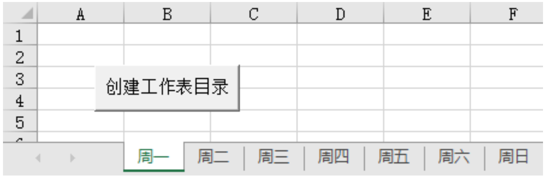


图 7–43　未创建目录的工作表



图 7–44　工作表目录

【语法补充】

（1）当删除非空工作表和合并非空单元格时都会弹出提示，从而影响代码的执行效率，所以执行这类操作前有必要通过“Application.DisplayAlerts = False”关闭提示。

（2）Sheets.Add 方法可以创建一个新工作表，它的返回值是一个工作表对象。如果仅仅创建工作表，那么可采用以下语句，表示在最左边创建一个工作表：

```
Sheets.Add before:=Sheets(1)
```

如果要对 Sheets.Add 创建的新表命名，那么 Sheets.Add 的参数必须使用括号：

```
Sheets.Add(before:=Sheets(1)).Name = "工作表目录"
```

（3）“{"编号", "目录"}”是公式中常用的常量数组表达式，而在 VBA 中要识别这类表达式需要添加方括号才行，所以代码中采用了“[{"编号", "目录"}]”。不过 Evaluate 也能实现同等功能，即“Evaluate("{1,2}")”。

（4）Hyperlinks.Add 表示在工作表中创建超链接，其语法如下：

```
Hyperlinks.Add(Anchor, Address, SubAddress, ScreenTip, TextToDisplay)
```

各参数的含义如表 7-12 所示。

表 7-12　Hyperlinks.Add的参数说明

名　称	必选/可选	说　明
Anchor	必选	超链接的位置。可为Range或Shape对象
Address	必选	超链接的地址，创建文件、网址和邮件链接时用
SubAddress	可选	超链接的子地址，创建工作簿内部链接时用
ScreenTip	可选	当鼠标指针停留在超链接上时所显示的屏幕提示
TextToDisplay	可选	要显示的超链接的文本

本例中参数 Anchor 赋值为 Cells(i, 2)，表示链接产生在 Cells(i, 2)单元格；参数 Address 赋值为空文本，同时对 SubAddress 参数赋值为“"'" & Sheets(i).Name & "'" & "!A1"”，表示链接到本工作簿中的 Sheets(i)工作表中的 A1 单元格，变量 i 的值随循环语句而变化，超链接的目标也相应地变化；参数 TextToDisplay 赋值为 Sheets(i).Name 表示在单元格中显示链接目标的工作表名称，可以随意指定文本；参数 ScreenTip 赋值为“"单击打开：" & Sheets(i).Name”表示光标停在单元格时产生的提示信息。

（5）ActiveWindow.FreezePanes 表示当前活动窗口是否被冻结，它是一个可读、可写的属性，如果对其赋值为 True 则表示冻结窗格，如果赋值为 False 则表示取消冻结。冻结的方式由活动单元格的位置决定。

通过读取 ActiveWindow.FreezePanes 的属性值也可用于判断窗口是否处于冻结状态。

本例案例文件请参考：..\ 第 7 章\ 7-22 创建工作表目录.xlsm

7.4 工作簿对象

在 VBA 中，Workbook 表示工作簿对象，而 Workbooks 则表示工作簿对象集合。

掌握工作簿的属性和方法与掌握工作表的属性和方法同等重要，本节提供 6 个与工作簿相关的案例。

7.4.1 打开带密码且带有自动宏的工作簿

【案例要求】如果知道工作簿的密码，要求用 VBA 打开该工作簿，不弹出录入密码的对话框。同时由于该工作簿中存在自动启动的宏，所以要求禁止运行该工作簿中的宏，只需要打开工作簿即可。

【知识要点】Workbooks.Open。

【程序代码】

```
Sub 打开带密码且带有自动宏的工作簿()'放置位置：模块中
    '禁用打开的所有文件中的所有宏，而不显示任何安全警告。
    Application.AutomationSecurity = msoAutomationSecurityForceDisable
    '打开工作簿时自动输入密码
    Workbooks.Open Filename:="C:\生产表.xlsm", Password:="ABCC"
    '恢复宏的处理方式：用“安全性”对话框中指定的安全设置来处理。
    Application.AutomationSecurity = msoAutomationSecurityByUI
End Sub
```

假设工作簿“生产表.xlsm”设置了打开密码“ABBC”，工作簿中有名为“Auto_open”的自动宏，而且工作簿保存在 C 盘中，那么执行以上过程后可以打开该工作簿，不会弹出录入密码的对话框，也不会执行其中的过程“Auto_open”。

【语法补充】

（1）Application.AutomationSecurity 属性用于控制在打开工作簿时对工作簿中的代码的处理方式，当赋值为 msoAutomationSecurityForceDisable 时表示禁止工作簿中的代码运行（仅针对用代码打开的工作簿中的代码），如果赋值为 msoAutomationSecurityByUI，则表示采用宏安全性对话框中的设置，这是默认的设置。

（2）Workbooks.Open 表示打开工作簿，其语法如下：

```
Workbooks.Open(FileName, UpdateLinks, ReadOnly, Format, Password, WriteResPassword, IgnoreReadOnlyRecommended, Origin, Delimiter, Editable, Notify, Converter, AddToMru, Local, CorruptLoad)
```

Workbooks.Open 的参数较多，其中最常用的参数是 FileName、UpdateLinks 和 Password，FileName 表示被打开的工作簿的名称及路径；UpdateLinks 表示假设工作簿中有外部链接时，打开工作簿是否更新该链接的值；Password 表示使用预设的密码打开工作簿。

其他参数的具体含义请查阅以下网址：

https://msdn.microsoft.com/zh-cn/library/microsoft.office.interop.excel.workbooks.open(v=office.15).aspx

本例案例文件请参考：..\ 第 7 章\ 7-23 打开带密码且带有自动宏的工作簿.xlsm

7.4.2 另存工作簿且以今天的日期命名

【案例要求】将工作簿另存到“D:\生产表”路径下，以今天的日期作为文件名称。

【知识要点】Workbook.SaveAs。

【程序代码】

```
    Sub 以今天日期保护工作簿() '放置位置：模块中
        '如果保存在“D:\生产表\”这个文件夹中
         If Len(Dir("D:\生产表\", vbDirectory)) > 0 Then
            '将活动工作簿另存为启用宏的文件格式，以今天日期作为文件名称
            ActiveWorkbook.SaveAs "D:\生产表\" & Format(Date, "yyyy-mm-dd") & ".xlsm",
xlOpenXMLWorkbookMacroEnabled
        End If
    End Sub
```

执行以上代码，如果路径“D:\生产表”存在，那么活动工作簿将自动保存到该路径中，文件名为当前的系统日期。如果文件夹不存在，则不会有任何反应。

【语法补充】

（1）Dir 函数的第二参数表示文件属性，默认值是 vbNormal。本例中要获取的是文件夹名称，所以第二参数采用“vbDirectory”。

（2） Workbook.SaveAs 表示另存工作簿，其语法如下：

```
    Workbook.SaveAs(FileName, FileFormat, Password, WriteResPassword, ReadOnlyRecommended,
CreateBackup, AccessMode, ConflictResolution, AddToMru, TextCodepage, TextVisualLayout, Local)
```

其中参数 FileName 表示保存后的文件名称，包含完整的路径；参数 FileFormat 表示文件格式，需要与参数 FileName 中的后缀名保持一致。例如 xlOpenXMLWorkbookMacroEnabled 对应 xlsm 格式，xlOpenXMLWorkbook 对应 xlsx 格式，xlAddIn 对应 xlam 格式，xlAddIn8 对应 xla 格式，xlExcel8 对应 xls 格式……

参数 Password 表示文件的打开密码；参数 CreateBackup 表示是否对文件创建备份，如果赋值为 True，则表示保存时自动创建一个备份文件，后缀名为 xlk。其余格式代码与后缀名的对应关系请参阅 VBA 帮助，查询帮助的关键字为“XlFileFormat 枚举”。

（3） Date 函数用于产生当前系统日期，其格式由系统的控制面板中的设置决定，所以有可能是“yyyy-mm-dd”，也可能是“mm/dd/yyyy”。由于文件名称中不能出现“/”，所以必须通过 Format 函数将 Date 产生的日期强制转换成以“-”作为分隔符的形式。

（4）本例采用日期作为名称，事实上也可以采用某个单元格的字符串或者某个工作表的名称作为工作簿的名称，读者可以试着修改本例代码去实现这类需求。

本例案例文件请参考：..\第 7 章\7-24 以今天日期保存工作簿.xlsm

7.4.3 将外部链接转换成值

【案例要求】工作簿中的公式引用了其他工作簿的数据，而且具有这类外部链接的单元格个数不确定。现要求一键实现转换所有公式为数值，从而断开外部链接。

【知识要点】Workbook.BreakLink 和 Workbook.LinkSources。

【程序代码】

```
    Sub 中断链接()                                                  '放置位置：模块中
        Dim aLinks                                                  '声明变量
        '获取工作簿中的所有链接到其他工作簿的外部链接
        aLinks = ActiveWorkbook.LinkSources(xlExcelLinks)
          If Not IsEmpty(aLinks) Then                               '如果有外部链接
            For i = 1 To UBound(aLinks)                             '遍历所有外部链接
                ActiveWorkbook.BreakLink   aLinks(i), xlLinkTypeExcelLinks   '中断链接
```

```
            Next i
        End If
    End Sub
```

执行以上代码，如果工作簿中没有链接到其他工作簿的公式，那么不会有任何反应。如果工作簿中存在若干个链接到其他工作簿的公式，那么可以瞬间中断这些链接。

【语法补充】

（1） Workbook.BreakLink 表示将链接到其他 Excel 工作簿或 OLE 源的公式转换为值，其语法如下：

```
Workbook.BreakLink(Name, Type)
```

Name 参数表示链接的名称，例如当外部链接是链接到其他工作簿中的单元格时，那么对 Name 赋值时就用工作簿名称。Type 参数表示链接类型，其可选值包括 xlLinkTypeExcelLinks 和 xlLinkTypeOLELinks，前者代表工作簿链接，后者表示 OLE 链接。

（2）Workbook.LinkSources 是一个包含工作簿中所有链接的数组。数组中包括链接的文档名、版本名、DDE 或 OLE 服务器名。如果工作簿中无链接，则返回 Empty。

由于本例中 Workbook.BreakLink 的第一参数需要使用外部链接的工作簿名称，而 Workbook.LinkSources 的返回值的第一个元素就是工作簿名称，所以两者配合即可中断第一个外部链接。再加上循环语句，即可中断所有链接到其他工作簿的外部链接。

本例案例文件请参考：..\ 第 7 章\ 7-25 将外部链接转换成值.xlsm

7.4.4 关闭工作簿且不保存修改内容

【案例要求】有些情况下需要打开工作簿提取其中的部分数据，因此希望关闭工作簿时不弹出“是否保存对文件的更改”的对话框。用 VBA 如何实现该需求呢?

【知识要点】Workbook.Close 和 Workbooks.Close。

【程序代码】

```
Sub 关闭活动工作簿且不提示保存()              '放置位置：模块中
    ActiveWorkbook.Close False                 '关闭活动工作簿且不保存
End Sub
```

执行以上过程可以关闭活动工作簿，并且不管工作簿是否修改过都会直接关闭，不会弹出“是否保存对文件的更改”的对话框。

如果打开了多个工作簿，需要一并关闭，那么可采用以下代码：

```
Sub 关闭所有工作簿且不提示保存()              '放置位置：模块中
    Dim wb As Workbook                        '声明变量
    For Each wb In Workbooks                  '遍历所有打开的工作簿
        wb.Close False                        '关闭工作簿且不保存
    Next wb
End Sub
```

也可以一次性关闭所有工作簿：

```
Sub 关闭所有工作簿且不提示保存 2()            '放置位置：模块中
    Application.DisplayAlerts = False         '关闭提示
    Workbooks.Close                           '关闭所有工作簿
    Application.DisplayAlerts = True          '恢复提示
End Sub
```

不过，关闭工作簿后 Excel 应用程序仍然处于打开状态，如图 7-45 所示。

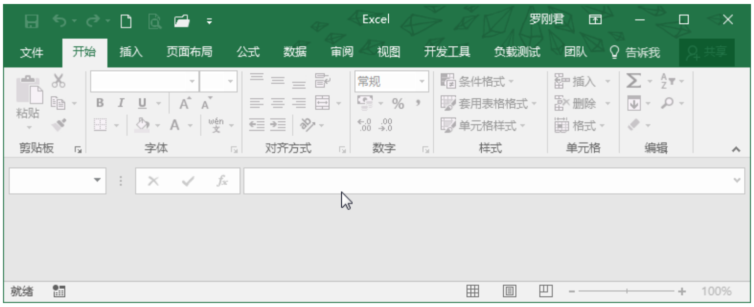


图 7–45　关闭工作簿但不关闭应用程序的状态

如果需要将应用程序一并关闭，那么可以采用以下代码：

```
Sub 关闭 Excel 应用程序不保存工作簿()                    '放置位置：模块中
    Application.DisplayAlerts = False                    '关闭提示
    Application.Quit                                     '退出应用程序
    Workbooks.Close                                      '关闭所有工作簿
    Application.DisplayAlerts = True                     '恢复提示
End Sub
```

【语法补充】

（1）Workbook.Close 表示关闭一个工作簿，语法如下：

```
Workbook.Close(SaveChanges, Filename, RouteWorkbook)
```

其中参数 SaveChanges 表示关闭时是否保存工作簿，赋值为 True 时表示保存，赋值为 False 时表示不保存，

参数 Filename 表示文件名，包含路径，通常需要另存时才使用此参数。例如活动工作簿名称为“123.xlsm”，如果需要将修改后的文件另存为 C 盘下的“456.xlsm”，那么可采用以下代码：

```
Sub 关闭活动工作簿()
    ActiveWorkbook.Close True, "C:\456.xlsm"             '关闭活动工作簿，同时另存时“456.xlsm”
End Sub
```

执行代码后，原文件“123.xlsm”的内容不变，修改后的内容保存在“456.xlsm”中。

参数 RouteWorkbook 表示是否将文件以邮件形式发送给某人，若不需要发送则忽略参数。

（2） Workbooks.Close 表示一次性关闭所有打开的工作簿，它没有参数，所以当工作簿在上次保存后有更改时会弹出是否保存对文件的更改的对话框。为了避免弹出对话框，在关闭工作簿语句之前加上关闭提示语句即可，即使用“Application.DisplayAlerts = False”。

事实上也可以将工作簿的 Saved 属性修改为 True 后再关闭工作簿，同样能屏蔽对话框。

（3） Workbooks.Close 表示关闭工作簿，而 Application.Quit 表示关闭应用程序，即 Excel 界面和工作簿是不同的概念。不过如果不关闭工作簿而直接关闭应用程序的话，将会弹出“是否保存工作簿”的对话框（假设工作簿有修改），所以需要配合 Workbooks.Close 使用。

本例案例文件请参考：..\ 第 7 章\ 7-26 关闭工作簿且不保存修改内容.xlsm

7.4.5　定时保存且备份工作簿

【案例要求】让活动工作簿每 5 分钟备份一次，即使最后活动工作簿自身未保存或者被破坏、误删除，仍然可以通过备份的文件找回数据。

【知识要点】Workbook.SaveCopyAs。

【程序代码】

```
Sub auto_open()      '放置位置：模块中
        Application.OnTime Now + TimeValue("00:05:00"), "auto_open"    '5 分钟后再执行本过程
      '备份活动工作簿到 C 盘中，命名为“生产表备份.xlsm”
        ActiveWorkbook.SaveCopyAs "C:\生产表备份.xlsm"
End Sub
```

以上代码可以在打开工作簿时自动执行。

不过在新建的工作簿中录入代码后需要按【F5】键手动执行过程，执行过程后先备份一次工作簿，5 分钟过后再备份一次。

【语法补充】

（1）Application.OnTime 用于安排一个在将来的特定时间运行的过程，既可以是具体指定的某个时间，也可以是指定的一段时间之后。

例如在 12 点钟执行过程 A，那么可使用以下代码：

```
Application.OnTime   TimeValue("12:00:00"), "A"
```

如果要在 10 秒后执行过程 B，那么可使用以下代码：

```
Application.OnTime   Now+TimeValue("00:00:10"), "B"
```

Application.OnTime 的语法如下：

```
Application.OnTime (EarliestTime, Procedure, LatestTime, Schedule)
```

Application.OnTime 各参数的含义如表 7-13 所示。

表 7-13　Application.OnTime的参数说明

名　称	必选/可选	说　明
EarliestTime	必选	希望此过程运行的时间，可以是指定的具体时间点，也可以当前时间之后的某一段时间
Procedure	必选	要运行的过程名称
LatestTime	可选	过程开始运行的最晚时间。例如，如果LatestTime参数设置为 EarliestTime + 30，且当到达 EarliestTime 时间时，由于其他过程处于运行状态而导致Excel不能处于“就绪”、“复制”、“剪切”或“查找”模式，那么Excel将等待30秒让第一个过程先完成。如果 Microsoft Excel 不能在 30 秒内回到“就绪”模式，则不运行此过程
Schedule	可选	如果为True，则表示指定一个新的计划任务；如果为False，则清除先前设置的过程。默认值为True

（2）Workbook.SaveCopyAs 表示将指定工作簿的副本保存到另一个文件，但不修改当前打开的工作簿。换言之，Workbook.SaveCopyAs 只保存副本，原文件并没有保存。例如有一个未保存的工作簿“工作簿 2”，当使用 Workbook.SaveCopyAs 将文件备份到“123.xlsx”后，“工作簿 2”本身仍然处于未保存状态。所以 Workbook.SaveCopyAs 和 Workbook.SaveAs 有较大的不同。

本例案例文件请参考：..\ 第 7 章\ 7-27 定时备份工作簿.xlsm

7.4.6　重命名活动工作簿

【案例要求】对活动工作簿执行命名，允许用户随意指定工作簿名称。

【知识要点】Workbook.Name、Workbook.FullName 和 Workbook.SaveAs。

【程序代码】

```
Sub 重命名活动工作簿()   '放置位置：模块中
      '如果利用 Dir 提取的活动工作簿的名称长度为 0（即未保存），那么提示用户，然后退出程序
```

```
        If Len(Dir(ActiveWorkbook.FullName)) = 0 Then MsgBox "请先保存工作簿", vbOKOnly, "友情提示": Exit
Sub
        Dim 原名称 As String, 新名称 As String, 后缀名 As String, 路径 As String   '声明变量
       原名称 = ActiveWorkbook.Name                           '提取活动工作簿名称
      '提取活动工作簿的后缀名
       后缀名 = StrReverse(Mid(StrReverse(原名称), 1, InStr(StrReverse(原名称), ".")))
      '弹出输入框让用户输入新的名称
       新名称 = Application.InputBox("请输入文件名", "新名称", Replace(原名称, 后缀名, "-2"), , , , , 2)
       If 新名称 = "False" Then End                           '如果单击了“取消”按钮则结束过程
       路径 = Replace(ActiveWorkbook.FullName, 原名称, "")     '提取活动工作簿的路径
       On Error Resume Next                                   '当有错误时继续执行
        '在 C 盘创建一个同名的文件夹（测试字符串能否作为文件名称）
        MkDir "C:\" & 新名称
        If Err <> 0 Then                                      '如果有错误
            MsgBox "您输入的字符不允许作为文件名，请重新输入！", vbOKOnly, "友情提示" '提示
        Else                                                  '否则
            RmDir "C:\" & 新名称                              '删除创建的文件夹
    '将活动工作簿另存为指定的名称与原文件路径相同
            ActiveWorkbook.SaveAs 路径 & 新名称 & 后缀名
            Kill 路径 & 原名称                                 '删除原来的文件
        End If
    End Sub
```

如果在工作簿未保存时执行以上过程将弹出提示框，然后自动结束过程。如果工作簿已经保存过，其完整名称为“D:\生产表\汇总表.xlsm”，那么执行此过程后将弹出如图 7-46 所示的对话框，供用户指定新的工作簿名称。

在此对话框中，VBA 代码会自动识别文件的原名称，在它后面加上“-2”作为默认的新名称。当然，也可以随意指定新的名称，例如“本月总表”，然后程序会将文件更名为“本月总表.xlsm”，其后缀名取自重命名之前的后缀名。

不过如果输入“/”、“\”、“?”等非法字符将弹出如图 7-47 所示的息信框，然后自动结束过程。



图 7-46　默认的新名称

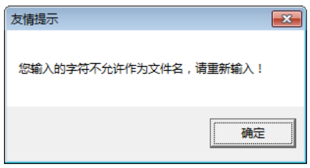


图 7-47　录入非法字符时弹出的信息框

【语法补充】

（1）Workbook.Name 属性表示工作簿的名称，不包含路径。如果工作簿未保存，那么此名称不包含后缀名，通常为“Book1”、“工作簿 1”，这取决于 Office 的语言版本。如果工作簿已保存过，那么 Workbook.Name 代表工作簿的名称，包含后缀名。

（2）Workbook.FullName 属性表示工作簿的全名，包含路径与后缀名，例如“D:\生产表\5 月.xlsm”。不过如果工作簿未保存则无法获取其路径与后缀名，此时它等同于 Workbook.Name。鉴于此特性，判断工作簿是否保存也可以采用以下思路：

```
    MsgBox IIf(ActiveWorkbook.FullName = ActiveWorkbook.Name, "未保存", "已保存")
```

（3）StrReverse 函数用于反转字符串，将一个字符串倒序显示，例如将 123 转换成为 321。

本过程中需要计算文件的后缀名，所以将 Workbook.Name 反转过来，然后通过 InStr 函数计算小圆点的位置，第一位到该小圆点前一位的所有字符即为文件的后缀名。

InStr 函数用于计算一个字符在字符串中首次出现的位置，其语法如下：

```
InStr([start, ]string1, string2[, compare])
```

其中各参数的含义如表 7-14 所示。

表 7-14　InStr的参数说明

部　分	可选/必选	说　明
start	可选参数	一个表示搜索起点的数值表达式。如果省略将从第一个字符的位置开始。如果指定了compare参数，则不能省略start参数
string1	必选参数	接受搜索的字符串表达式
string2	必选参数	被搜索的字符串表达式
Compare	可选参数	指定字符串比较时是否区分大小写，赋值为vbBinaryCompare时表示要区分大小写，赋值为vbTextCompare时不区分大小写

简单而言，InStr 函数就是从第 start 个字符开始，在 string1 中查找 string2，返回首次出现的位置，当查找不到时返回 0。如果 Compare 值为 vbBinaryCompare，则表示查找字母时区分大小写。

MsgBox InStr(1, "Aba1", "a", vbBinaryCompare)——返回值为 3。

MsgBox InStr(1, "Aba1", "a", vbTextCompare) ——返回值为 1。

MsgBox InStr("Aba1", "a")——返回值为 3，即表示在忽略可选参数时要区分大小写。

本例中“StrReverse(Mid(StrReverse(原名称), 1, InStr(StrReverse(原名称), ".")))”表示将文件名称反转后查找“.”的首次出现位置，然后利用 Mid 函数从第一位开始取值，到“.”的位置结束，最后将取出的字符再次反转得到文件的后缀名。

（4）Windows 对于文件名有一些限制，包括长度限制和字符限制。其中长度限制是指文件名称（包含路径）不能超过 255 个字符，而字符限制则是指文件名称中不能使用非法字符，例如“/”、“\”、“?”等符号。

所以本例为了避免出错，先用输入的字符去创建一个文件夹，如果创建不成功则说明输入的字符过长或者包含非法字符；如果成功则马上将活动工作簿另存，将该名称作为文件的新名称，最后删除原来的文件和作为测试用途的文件夹即可。

Excel 不支持对打开的工作簿直接重命名，VBA 同样绕不过这道坎。不过由于 VBA 的执行速度较快，测试字符串的合法性、另存工作簿、删除原文件和文件夹等操作都可以瞬间完成，所以可以采用本例的思路来实现重命名，和直接重命名文件在效率上没有差异，对于用户而言也完全没有烦琐的感觉。

（5）MkDir 语句可以创建指定名称的文件夹，RmDir 语句则用于删除指定名称的文件夹。

（6）由于变量“路径”被声明为 String，所以当单击“取消”按钮时 Application.InputBox 的返回值为“False”，程序会把文件重命名为“False.xlsx”或者“False.xlsx”，所以有必要通过 If 语句判断用户是否单击了“取消”按钮。不过此思路也有一个漏洞——假设用户手动指定“False”也会执行不成功，好在实际工作中通常没有人会将 Excel 文件命名为“False”。

本例案例文件请参考：..\ 第 7 章\ 7-28 对活动工作簿重命名.xlsm

7.5　课后思考

1．在活动工作簿的所有工作表中查找包含“迟到”的所有单元格，统计符合条件的单元格的个数。请用循环语句加 Range.Find 方法实现。

2．删除打开的所有工作簿的所有工作表中的图形对象。

3．用红圈标示成绩表中倒数三名同学的成绩所在单元格。

4．假设在工作表中有多个合并单元格，要求将所有合并单元格取消合并，且让取消合并后的每个单元格填充合并的值。效果如图 7-48 所示，A1:F4 为取消合并前的效果，A6:F9 为取消合并后的效果。

<table>
<tr><th></th><th>A</th><th>B</th><th>C</th><th>D</th><th>E</th><th>F</th></tr>
<tr><td>1</td><td rowspan="4">山东省</td><td>烟台市</td><td rowspan="4">福建省</td><td>漳州市</td><td rowspan="4">云南省</td><td>昭通市</td></tr>
<tr><td>2</td><td>淄博市</td><td>厦门市</td><td>玉溪市</td></tr>
<tr><td>3</td><td>枣庄市</td><td>三明市</td><td>曲靖市</td></tr>
<tr><td>4</td><td>威海市</td><td>泉州市</td><td>丽江市</td></tr>
<tr><td>5</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>6</td><td>山东省</td><td>烟台市</td><td>福建省</td><td>漳州市</td><td>云南省</td><td>昭通市</td></tr>
<tr><td>7</td><td>山东省</td><td>淄博市</td><td>福建省</td><td>厦门市</td><td>云南省</td><td>玉溪市</td></tr>
<tr><td>8</td><td>山东省</td><td>枣庄市</td><td>福建省</td><td>三明市</td><td>云南省</td><td>曲靖市</td></tr>
<tr><td>9</td><td>山东省</td><td>威海市</td><td>福建省</td><td>泉州市</td><td>云南省</td><td>丽江市</td></tr>
</table>

图 7-48　取消合并单元格前后效果比较

5．假设工作表中有多个批注，要求在每个批注的文本前插入用户名称，VBA 获取用户名称的方法是 Application.UserName。

第 8 章　深入剖析 VBA 的各种事件

事件的价值在于让代码按预设的条件自动执行，从而解脱双手，提高工作效率。

事件包含工作表事件、工作簿事件、Excel 应用程序事件、图表事件、窗体事件和各种控件的事件。关于窗体的事件将在本书第 12 章中详细阐述，本章重点展示工作表事件、工作簿事件和 Excel 应用程序事件，其中工作表事件和工作簿事件更是重中之重。

本章要点

◆ 事件的级别与顺序

◆ 禁用与启用事件

◆ 工作表事件详解

◆ 工作簿事件详解

◆ 应用程序事件详解

8.1　事件的级别与顺序

本章的重点在于工作表事件、工作簿事件和应用程序事件，它们三者之间是相通的，只要学会一个即可触类旁通。不过这三类事件之间也存在执行顺序的差异和层级关系，了解这些差异能更好地应用事件，懂得在什么需求下应选用什么事件。

8.1.1　事件的级别与代码保存位置

工作表级事件、工作簿级事件和应用程序级事件三者之间遵循以下层级关系：

工作表级事件＜工作簿级事件＜应用程序级事件

Excel 2016 有 17 类工作表级事件，读者可以打开以下网址，查看工作表级事件的具体说明。

https://msdn.microsoft.com/zh-cn/library/ff841127

Excel 2016 有 40 类工作簿级事件，由于工作簿事件处于工作表事件的上层，所以它包含了所有工作表事件，换言之，所有工作表事件都可以用工作簿事件来完成。例如工作表事件“Worksheet_Activate”也可以代之以工作簿事件“Workbook_SheetActivate”，任何触发“Worksheet_Activate”事件的操作必定会触发“Workbook_SheetActivate”事件。

读者可以打开以下网址，查看工作簿事件的具体说明。

https://msdn.microsoft.com/zh-cn/library/ff839916.aspx

Excel 2016 有 47 类应用程序级事件，它处于事件的最顶层，同时包含了工作表事件、工作簿事件和应用程序自身所特有的事件。

读者可以打开以下网址，查看应用程序级事件的具体说明。

https://msdn.microsoft.com/zh-cn/library/ff198091.aspx

工作表级事件仅仅作用于单个工作表及工作表中的单元格，代码保存在指定名称的工作表的事件代码窗口中。图 8-1 即为一个作用于 Sheet1 的工作表事件，选择 Sheet1 工作表中的任意单元格时将执行此事件过程中的命令，并且在状态栏会产生“这是工作表事件！”的提示信息。

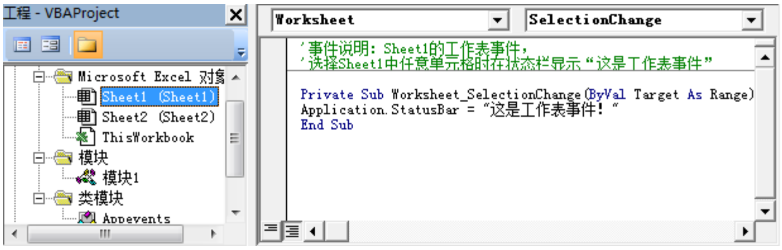


图 8–1　工作表事件

工作簿级事件对整个工作簿都生效，工作簿中任意单元格或者任意工作表都有可能触发工作簿级事件，工作簿级事件的代码保存在 ThisWorkbook 窗口中。在图 8-2 中包含了一个工作簿级事件，修改此工作簿中的任意单元格都将执行此事件过程中的命令。



图 8–2　工作簿事件

应用程序级事件作用于所有打开的工作簿，即任意工作簿中的任意工作表都会触发应用程序级事件，应用程序级事件的代码保存在 ThisWorkbook 窗口或者类模块中，不过也需要配合模块中的代码一同使用。

图 8-3 中包含了一个应用程序级事件，当新建工作簿时将执行此事件过程中的命令。

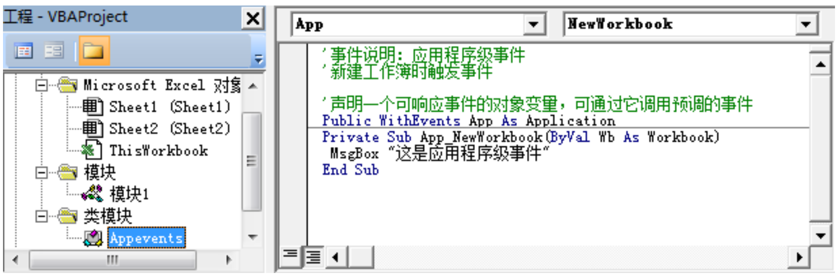


图 8–3　应用程序级事件

此处仅说明三类事件的保存位置，代码的书写方式和含义将在本章后面几节分别阐述。读者可以打开随书案例文件，分别测试三个图片中所讲的三种操作，验证三个事件的命令是否能正常工作。

本例案例文件请参考：..\ 第 8 章\ 8-1　三种事件代码的保存位置演示.xlsm

8.1.2 事件的执行方式

事件是 VBA 预先内置在应用程序中的过程，其过程名称已经固定不可修改。不过可以在事件过程中随意编写代码，使触发事件时自动执行程序员所指定的命令。

区分事件过程和普通过程的方法有两个：一是看代码的位置，普通过程可以放在工作表事件代码窗口、工作簿事件代码窗口和任意模块中，而事件过程只能放在指定的位置才能使用，或者说放在正确的位置后事件过程才具备事件的特性，否则就成了普通过程；二是看过程名称的命名方式，由于事件过程是内置的过程，其名称必须遵循内部的规则，所以可以通过名称判断过程是否为事件过程。

事件过程的名称包括三项内容：事件的级别、触发事件的动作和触发事件的对象。

例如，“Worksheet_SelectionChange”事件的完整名称如下：

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)

End Sub
```

其中 Worksheet 部分表示事件的级别——工作表事件；SelectionChange 部分表示触发事件的动作——选区发生变化时触发事件（亦可理解为重选区域）；Target 部分表示触发事件的对象，它代表当前选择的单元格对象。例如在 Sheet2 中选择 A1:A10 区域时触发了“Worksheet_SelectionChange”事件，那么 Target 则代表 A1:A10 区域。

也有例外情况存在，例如“Workbook_Open”事件，此事件过程不含参数，即没有触发事件的对象，只有动作，表示打开工作簿时执行此事件过程。

```
Private Sub Workbook_Open()

End Sub
```

换言之，事件的名称已经完整地表达出事件的执行方式，看到名称即可明白它是什么级别的事件，在执行某种操作时会触发该事件。

下面提供更多的事件名称分析。

```
Private Sub Worksheet_Calculate()

End Sub
```

上述代码表示一个工作表级事件，当工作表中任意单元格的公式执行计算时触发事件。

```
Private Sub Workbook_WindowActivate(ByVal Wn As Window)

End Sub
```

上述代码表示一个工作簿级事件，激活工作簿窗口时触发事件，参数 Wn 代表窗口对象。

```
Private Sub Workbook_SheetChange(ByVal Sh As Object, ByVal Target As Range)

End Sub
```

上述代码表示一个工作簿级事件，工作簿中任意工作表的任意单元格的值发生变化时触发此事件，参数 Sh 代表工作表，Target 代表单元格或区域。

```
Private Sub Worksheet_Change(ByVal Target As Range)

End Sub
```

上述代码表示一个工作表级事件，修改代码所在的工作表单元格时触发此事件，参数 Target 代表被修改数据的单元格对象。

8.1.3 事件的执行顺序

事件的执行顺序与层级关系相反，遵循从低到高的规则，即当一个操作同时触发工作表级事件、工作簿级事件和应用程序级事件时，那么最先执行的是工作表级事件，接着是工作簿级事件，最后是应用程序级事件。

在实际工作中，同时使用以上三种级别的事件的机率太小，而同时使用多个同级别事件的机率则相当大。当一个操作触发了多个同级别的事件时，它们的执行顺序如何呢？

例如活动工作表是Sheet1时，单击工作表Sheet2将触发Sheet1的“Worksheet_ Deactivate”事件，然后是 Sheet2 的“Worksheet_Activate”事件。可以通过以下步骤验证。

（1）在工程资源管理器中双击 Sheet1，然后在右边的代码窗口中录入代码：

```
Private Sub Worksheet_Deactivate()      '放置位置：Sheet1 工作表事件代码窗口
    MsgBox "Worksheet_Deactivate 事件"
End Sub
```

（2）双击 Sheet2，然后在右边的代码窗口中录入代码：

```
Private Sub Worksheet_Activate()      '放置位置：Sheet2 工作表事件代码窗口
    MsgBox "Worksheet_Activate 事件"
End Sub
```

（3）返回工作表界面，单击工作表名称 Sheet2，然后将弹出两次提示框，首先提示“Worksheet_Activate 事件”，然后是“Worksheet_Deactivate 事件”。

可以使用同样方法测试工作簿事件“Workbook_NewSheet”、“Workbook_SheetActivate”和“Workbook_SheetDeactivate”的先后顺序。

本例案例文件请参考：..\ 第 8 章\ 8-2 测试工作表事件的执行顺序.xlsm

8.2 禁用与启用事件

事件是由预先指定的某些操作触发的，当编写好事件过程的代码后，只要符合条件就会触发事件。然而有些时候并不需要事件过程继续执行，而是要求临时禁止触发事件，而在符合另外的条件时需要恢复事件的触发机制，这就涉及到启用与禁用事件。

8.2.1 临时关闭事件

只要编写了事件过程代码，那么符合条件时就会触发事件。如果需要彻底禁止事件只有删除事件过程的代码或者将代码转换成注释。

然而如果只是临时性地禁止事件，当需要时再启用事件，则通过代码来关闭事件更具人性化。

禁用和启用事件可以通过修改 Application.EnableEvents 属性值来实现，当赋值为 True 时表示启用事件，当赋值为 False 时表示禁用事件。

```
Sub 禁用事件()  '放置位置：模块中
    Application.EnableEvents = False
End Sub
Sub 启用事件()  '放置位置：模块中
    Application.EnableEvents = True
End Sub
```

当执行过程“禁用事件”后，工作簿中的任何操作都将无法触发事件，直到执行过程“启用事件”或者重启 Excel。

8.2.2 防止事件的连锁反应

在 Excel 的事件过程中可以随意编写代码，可以执行任何操作，同时由于事件总是由指定的操作所触发，所以如果事件过程中的某句代码执行了可以触发当前事件的操作，那么就会产生事件的连锁反应。

所谓连锁反应是指触发事件时执行某个命令，而该命令又会继续触发事件，在相互作用下事件过程反复地执行下去，从而产生负面影响。

为了避免这种情况，可以在触发事件的命令之前将 Application.EnableEvents 属性赋值为 False，操作结束后再将该属性赋值为 True。

通过以下步骤可以了解事件的连锁反应。

（1）在工程资源管理器中双击工作表 Sheet1，然后在右边的代码窗口中录入以下代码：

```
Private Sub Worksheet_Change(ByVal Target As Range) '放置位置：Sheet1 工作表事件代码窗口
    If Target.Value < 60 Then
        Target = Target.Value & "（不及格）"
    Else
        Target = Target.Value & "（及格）"
    End If
End Sub
```

此代码是一个工作表级事件，当 Range 对象 Target 的值发生变化时触发事件，事件过程是如果 Target 单元格的值小于 60，则在该单元格的值后面标注“(不及格)”，否则标注“(及格)”。

（2）返回工作表界面，进入 Sheet1 工作表，随意选择单个单元格，然后在其中录入数值 50，当按下【Enter】键后该单元格的值不是变成“50（不及格）”，而是 50（不及格）（及格）”，这是为什么呢?

这是因为“Worksheet_Change”事件的触发条件是 Target 单元格的值产生变化，而本事件过程中的代码刚好是修改 Target 单元格的值，所以在触发事件时执行事件中的代码，而执行事件中的代码时又继续触发事件，从而造成事件的连锁反应，总共会触发事件 84 次（在 Excel 2010 中会触发 95 次，而 Excel 2016 中则触发 84 次，其他版本又有所不同）。

当第一次触发事件时，单元格的值为 50，由于它小于 60 所以在右边标注“(不及格)”；当标注“(不及格)”几个字时，这个操作触发了第二次事件，此时继续将单元格 Target 的值与 60 执行比较，由于字符串“50（不及格）”大于 60，所以在它右边标注“(及格)”……当重复触发 84 次后结束。

不过事件的连锁反应并非总是 84 次，例如下面的事件过程表示单元格的值发生变化时，在它右边的单元格中赋值为 Target 的值加 1。经测试后此事件会产生连锁反应，总共触发事件 50 次（在 Excel 2010 中会触发 57 次，而 Excel 2016 中则触发 50 次，其他版本又有所不同）。

```
Private Sub Worksheet_Change(ByVal Target As Range) '放置位置：Sheet1 工作表事件代码窗口
 Target(1).Offset(0, 1) = Target(1) + 1
End Sub
```

事实上，不仅不同事件的触发次数不同，就算完全相同的代码在不同版本的 Excel 中执行时也会有不同结果。不过重点不在于了解次数的多少，只要知道它会产生连锁反应并带来负面影响的这个事实即可，然后采用相应的补救措施。

对于以上两个过程，只要通过 EnableEvents 属性禁用和启用事件即可。修改后的代码如下：

```
Private Sub Worksheet_Change(ByVal Target As Range) '放置位置：Sheet1 工作表事件代码窗口
    Application.EnableEvents = False
    If Target(1).Value < 60 Then
        Target(1) = Target.Value & "（不及格）"
    Else
        Target(1) = Target.Value & "（及格）"
    End If
    Application.EnableEvents = True
End Sub
```

```
Private Sub Worksheet_Change(ByVal Target As Range) '放置位置：Sheet1 工作表事件代码窗口
    Application.EnableEvents = False
    Target(1).Offset(0, 1) = Target(1) + 1
    Application.EnableEvents = True
End Sub
```

修改后的代码可以正常运作，不再产生连锁反应，在任何情况下都只触发一次事件。

本例案例文件请参考：..\ 第 8 章\8-3 启用与禁用事件.xlsm

8.3　工作表事件详解

工作表事件是指操作工作表或工作表中的单元格时触发的事件，它处于工作表、工作簿和应用程序级事件中的底层。工作表事件的代码仅作用于代码所在工作表，对其他工作表无效。

8.3.1　工作表事件列表

在 Excel 2016 中，工作表级事件有 17 个，如表 8-1 所示。

表 8-1　Excel 2016 的工作表事件列表

工作表事件名称	说　明
Activate	激活Activate事件的代码所在的工作表时触发此事件
BeforeDelete	删除BeforeDelete事件过程所在的工作表后触发此事件
BeforeDoubleClick	当双击工作表中的单元格时触发此事件
BeforeRightClick	鼠标右键单击工作表中的单元格时触发此事件
Calculate	工作表中的公式执行重算之后触发此事件
Change	当更改工作表中的单元格，或外部链接引起单元格的更改时触发事件
Deactivate	激活其他工作表时触发此事件
FollowHyperlink	当单击工作表上的任意超链接时触发此事件
LensGalleryRenderComplete	在工作中使用快速分析时触发此事件
PivotTableAfterValueChange	在编辑或重新计算（针对包含公式的单元格）数据透视表中的单元格或单元格区域后触发此事件
PivotTableBeforeAllocateChanges	在向数据透视表应用更改前触发此事件
PivotTableBeforeCommitChanges	在针对 OLAP 数据源提交对数据透视表的更改前触发此事件
PivotTableBeforeDiscardChanges	在放弃对数据透视表所做的更改之前触发此事件
PivotTableChangeSync	在对数据透视表进行更改之后触发此事件

续表

工作表事件名称	说　　明
PivotTableUpdate	工作簿中的数据透视表更新后触发此事件
SelectionChange	在工作表上的选定区域发生改变时触发此事件
TableUpdate	连接到数据模型的查询表的工作表上更新后触发此事件

在工作表事件中常用事件包含 Activate、BeforeDoubleClick、BeforeRightClick、Change 和 SelectionChange 等。

要快速查询工作表支持哪些事件，只要从代码窗口上方的对象窗口中选择“Worksheet”，再单击右边的过程窗口即可，其下拉列表中提供了当前版本的 Excel 所支持的所有事件名称，如图 8-4 所示。

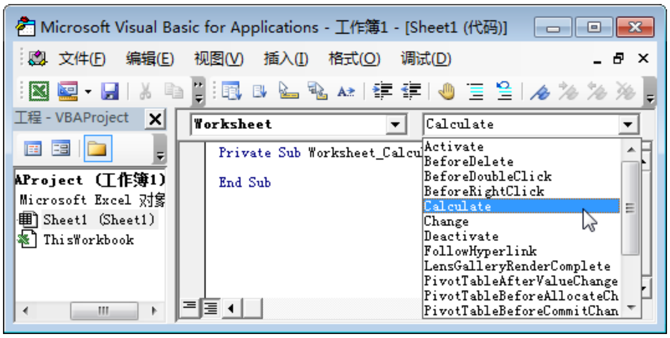


图 8–4　Excel 2016 的工作表事件过程列表

8.3.2　Change 事件的特例

工作表的 Change 和 SelectionChange 两个事件在工作中应用最频繁，其中后者比较容易理解，即改变选区时触发事件，但对于前者却很难准确地判断何时触发此事件。

工作表的 Change 事件全称是“Worksheet_Change”事件，表示在修改单元格的值时触发此事件，然而 Excel 内部对它的定义与大家按常理推断的结果可能会稍有不同，例如大家认为某个操作可能会触发 Change 事件，但是 Excel 却将它排除在外，也可能大家认为不会触发事件的操作事实上又触发了事件。下面列举七种比较容易混淆的特例。

1．删除空单元格时会触发工作表的 Change 事件

对于工作表的 Change 事件，准确的描述是“修改单元格时触发的事件”，而不是“单元格的值发生改变时触发的事件”。明白这一点很重要，利用这个概念判断删除空单元格时会不会触发工作表的 Change 事件就容易多了，因为删除空单元格这个操作正是修改单元格的值，符合触发事件的条件，至于单元格的值在修改前后是否产生变化并不重要。

2．插入批注时不触发工作表的 Change 事件

插入批注时不触发工作表的 Change 事件，这个其实不难理解。可以将批注理解为附加在单元格的图形对象，它并不隶属于单元格本身，添加批注时并没有修改单元格的值。

3．更新单元格格式后不触发工作表的 Change 事件

更新单元格的格式信息后，单元格中显示的值会发生变化，但是单元格本身的值是不变的。换言之，更新格式后只是改变显示效果，并没有修改单元格本身的值，所以不会触发事件。

4．用公式修改单元格的值时不触发工作表的 Change 事件

如果公式跨表引用其他工作表的单元格，被引用的值发生变化时，公式所在单元格的值也会相应地变化，但是这个过程只会触发 Calculate 事件，不会触发 Change 事件。

5．清除格式和超链接时触发工作表的 Change 事件

在对单元格设置格式和超链接时不会触发工作表的 Change 事件，但是在清除格式和超链接时却会触发事件，这不得不说是一个比较特殊的案例。

6．对数据分列时不会触发工作表的 Change 事件

在将一个单元格的数据通过“数据”选项卡中的“分列”拆分到多个单元格时，涉及的多个单元格的值都会产生变化，但是此操作并不会触发 Change 事件。

7．通过表单控件修改单元格的值时不触发工作表的 Change 事件

很多表单控件都可以修改单元格的值，例如组合框、列表框、数值调节按钮、单选按钮、复选框等。通过表单控件修改单元格的值时，不会触发任何工作表事件。

8.3.3　事件案例：激活工作表时验证访问权限

【案例要求】工作簿中有多个工作表，其中总表需要验证权限，在符合条件时才可以查看其中的数据。

【知识要点】Worksheet_Activate 事件。

【程序代码】

```
Private Sub Worksheet_Activate()   '工作表 Activate 事件（'放置位置：“总表”事件代码窗口）
    Dim Pass As String              '声明变量,String 型
Pass = Application.InputBox("请输入访问口令", "确认权限", , , , , , 2)  '弹出输入框，让用户录入口令
 '如果验证不成功则激活前一个工作表
  If Pass <> "VBA" Then Sheets(ActiveSheet.Index - 1).Activate
End Sub
```

假设工作簿中的工作表按图 8-5 的方式排列，以上事件过程的代码保存在总表的代码窗口中，那么在工作表界面单击总表时会弹出图如 8-6 所示的对话框。

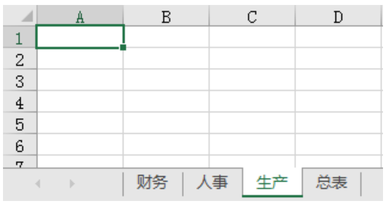


图 8–5　工作表布局



图 8–6　口令输入框

如果输入的口令等于“VBA”，那么过程自动结束，顺利进入总表中；如果输入的口令不等于“VBA”，那么将激活总表前面的一个工作表，从而阻止用户进入总表。

【知识补充】

（1）工作表事件代码仅对代码所在的工作表生效，所以代码的存放位置很重要。需要控制哪个工作表，代码就保存在哪个表的代码窗口中。

（2）“Worksheet_Activate”事件在激活工作表之后触发，所以在通过 Application.InputBox 弹出对话框时可以看到总表中的数据。不过如果在弹出对话框前激活上一个工作表就可以解决这

个问题，代码如下：

```
Private Sub Worksheet_Activate() '工作表 Activate 事件，代码放置位置：“总表”代码窗口
  Dim Pass As String, ShtName As String              '声明变量为 String 型
  ShtName = ActiveSheet.Name                          '记录活动工作表名称
  '如果活动工作表是第一个表则激活后一个工作表，否则激活前一个工作表
  Sheets(ActiveSheet.Index + IIf(ActiveSheet.Index = 1, 1, -1)).Activate
'弹出对话框，让用户录入口令
  Pass = Application.InputBox("请输入访问口令", "确认权限", , , , , , 2)
  Application.EnableEvents = False                    '禁用事件
'如果验证成功则进入 ShtName 代表的工作表
  If Pass = "VBA" Then Sheets(ShtName).Activate
  Application.EnableEvents = True                     '恢复事件
End Sub
```

修改后的代码可以避免用户输入正确口令前看到总表的数据，如果口令不正确则只能激活总表前一个工作表或者后一个工作表，口令正确则进入总表。不过由于事件是由激活总表所触发的，而事件过程中又有激活总表的命令，所以它们会产生连锁反应，因此代码中需要禁用事件。

（3）“总表”有可能排列在最右边，也可能排列在最左边，因此要用代码“IIf(ActiveSheet.Index = 1, 1, -1)”防错。当“总表”在最左边时，ActiveSheet.Index = 1。

（4）与权限验证相关的代码都需要保护起来，避免用户删除代码。选择菜单“工具”→“VBAProject”，然后进入“保护”选项卡，勾选 “查看时锁定工程”单选框，再在下方的两个文字框中录入密码即可。图 8-7 即为保护工程的密码录入界面。

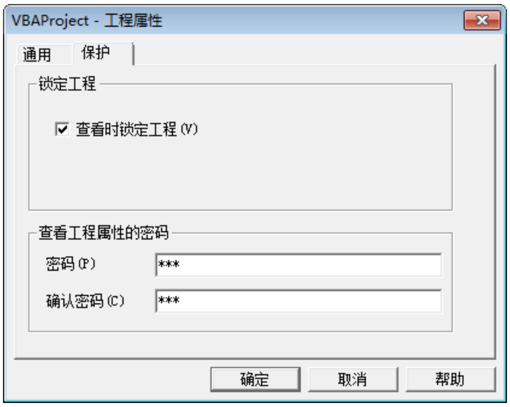


图 8–7　保护工程的密码录入界面

本例案例文件请参考：..\ 第 8 章\ 8-4 激活工作表时验证访问权限(VBA 密码 123).xlsm

8.3.4　事件案例：自动标示当前行的背景

【案例要求】鼠标光标单击哪一行就在哪一行添加条件格式，将该行用背景色突出显示。

【知识要点】Worksheet_SelectionChange 事件。

【程序代码】

```
'工作表 SelectionChange 事件（放置位置：“Sheet1”事件代码窗口）
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
  With Target.EntireRow                                     '引用选区的整行
    Cells.FormatConditions.Delete                           '删除所有单元格的条件格式
```

```
        .FormatConditions.Add Type:=xlExpression, Formula1:="=TRUE"      '添加新的条件格式
        .FormatConditions(1).Interior.Color = 65535                      '为条件格式指定背景色
    End With
End Sub
```

将以上代码放在 Sheet1 工作表的代码窗口中，然后返回工作表界面，在 Sheet1 工作表中随意单击单元格，那么选中单元格的所在行将会自动产生条件格式，使用黄色背景突出显示，效果如图 8-8 所示。

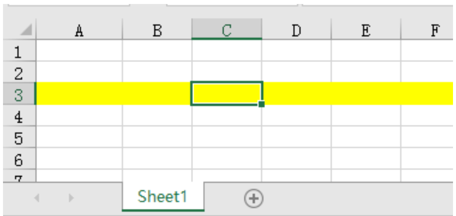


图 8–8　单击单元格时整行标示背景色

由于在事件过程中使用了“Cells.FormatConditions.Delete”，所以重选区域后上次产生的条件格式将自动消失，选中区域不会产生多个条件格式。

【知识补充】

（1）Worksheet_SelectionChange 事件表示代码所在工作表的选区变化时触发的事件，如果选区不变，只是活动单元格产生变化时，不会触发此事件。

参数 Target 代表当前选择的单元格或者区域，可以是多个区域。

（2）FormatConditions.Delete 表示删除所有条件格式，而 FormatConditions. Add 则表示创建新的条件格式，其完整语法如下：

```
FormatConditions.Add(Type, Operator, Formula1, Formula2)
```

其中各参数的含义如表 8-2 所示。

表 8-2　FormatConditions.Add的参数说明

名　称	必选/可选	说　明
Type	必选	指定条件格式是基于单元格的值还是基于表达式。如果是表达式则需要提供一个公式
Operator	可选	条件格式运算符。可为以下XlFormatConditionOperator 常量之一：xlBetween、xlEqual、xlGreater、xlGreaterEqual、xlLess、xlLessEqual、xlNotBetween 或 xlNotEqual。如果Type为xlExpression，则忽略 Operator 参数
Formula1	可选	与条件格式相关联的值或表达式。可为常量值、字符串值、单元格引用或公式
Formula2	可选	当Operator为xlBetween或xlNotBetween时，Formula2是与条件格式第二部分相关联的值或表达式（否则忽略该参数）。可为常量值、字符串值、单元格引用或公式

（3）代码“FormatConditions(1).Interior.Color = 65535”中的 FormatConditions(1)表示条件格式集合中的第一个条件格式，而“Interior.Color”则表示单元格内部的填充颜色，本例赋值为 65535，表示黄色。表 8-3 提供了常用颜色的编号。

表 8-3　常用颜色的编号一览

颜　色	编　号	颜　色	编　号
红色	255	紫色	10498160
蓝色	12611584	橙色	49407
绿色	5287936	黑色	0
黄色	65535	白色	16777215

本例案例文件请参考：..\ 第 8 章\ 8-5 自动标示当前行的背景.xlsm

8.3.5 事件案例：双击单元格时选中所有相同值

【案例要求】在如图 8-9 所示的成绩表中有若干数据，为了方便了解哪些人的成绩相同，现要求在双击任意单元格时同时选中所有相同值的单元格。

【知识要点】Worksheet_BeforeDoubleClick 事件。

【程序代码】

```
'工作表 BeforeDoubleClick 事件（放置位置： “Sheet1”事件代码窗口）
Private Sub Worksheet_BeforeDoubleClick(ByVal Target As Range, Cancel As Boolean)
  If Len(Target.Value) = 0 Then Exit Sub '如果被双击的单元格是空值则结束过程
'声明变量
  Dim 查找区域 As Range, 首个对象 As Range, 下个对象 As Range, 最终目标 As Range
  Set 查找区域 = ActiveSheet.UsedRange               '将本表的已用区域赋值给变量“查找区域”
'开始查找，查找对象是 Target 值，匹配方式为完全匹配
  Set 首个对象 = 查找区域.Find(Target, , xlValues, xlWhole)
  Set 下个对象 = 首个对象                            '将“下个对象”变量赋值为第一次找到的单元格
  Set 最终目标 = 首个对象                            '将“最终目标”变量赋值为第一次找到的单元格
  Do                                                 '使用 Do Loop 循环语句批量查找
    Set 下个对象 = 查找区域.FindNext(下个对象)       '查找下一个，从上次查找到的目标之后开始查找
    If 下个对象.Address = 首个对象.Address Then      '如果找到的下个目标和第一次找到的地址一样
        最终目标.Select                              '那么选择变量“最终目标”所代表的区域
        End                                          '结束一切，包括结束循环、结束过程、释放变量
    Else                                             '否则（表示发现了新目标）
        Set 最终目标 = Union(最终目标, 下个对象)     '将新找到的单元格与变量“最终目标”合并
    End If
  Loop
  最终目标.Select                                    '选择最终目标
End Sub
```

返回工作表界面，双击单元格 E4，由于 D6 和 B9 的也是 80，因此双击 E4 后程序会同时选中值为 80 的三个单元格，包含 E4、D6 和 B9，如图 8-9 所示。

E4 | 80

	A	B	C	D	E	F
1	姓名	语文	数学	体育	计算机	外语
2	计尚云	52	71	93	90	69
3	赵国	99	94	52	98	68
4	罗至贵	76	89	52	80	73
5	徐大鹏	65	81	83	63	64
6	张志坚	92	92	80	100	96
7	朱千文	61	85	99	62	77
8	赵秀文	55	100	84	50	79
9	梁爱国	80	55	90	64	52
10	梁兴	65	69	65	98	99
11	陈随机	70	64	58	58	82

图 8–9 双击时选中所有相同值所在单元格

如果双击 E1:E2 以外的单元格，那么事件过程会自动结束，不会有任何反应。

【知识补充】

（1）Worksheet_BeforeDoubleClick 事件表示在双击单元格时触发的事件，由于不可能同时双击多个单元格，所以参数 Target 永远只代表单个单元格。

参数 Cancel 用于控制事件过程中是否继续响应原本的双击操作。如果事件过程将此参数设

为 True，则在完成此过程后，不执行默认的双击操作。简而言之，此参数可以决定被双击的单元格是否进入编辑状态。通常忽略此参数，不需要赋值。

（2）工作表的双击事件对工作表中任意单元格都生效，但是由于有“If Len(Target.Value) = 0 Then Exit Sub '”，因此在双击空白单元格时不会有任何反应。

本例案例文件请参考：..\ 第 8 章\8-6 双击单元格时选中所有相同值的单元格.xlsm

8.3.6　事件案例：在特定区域右击单元格时产生工作表目录

【案例要求】右击 A 列或 C 列时可以弹出工作表目录，而单击其他列则保持原本的操作不变。

【知识要点】Worksheet_BeforeRightClick 事件。

【程序代码】

```
'工作表 BeforeRightClick 事件（'放置位置：“Sheet1”事件代码窗口）
Private Sub Worksheet_BeforeRightClick(ByVal Target As Range, Cancel As Boolean)
    If Intersect(Target, Range("A:A,C:C")) Is Nothing Then End   '如果与 A、C 列不重叠则结束过程
    Cancel = True                                               '取消原来的快捷菜单
    Application.CommandBars("Workbook tabs").ShowPopup          '调用系统自带的工作表目录
End Sub
```

将以上代码保存在 Sheet1 工作表的代码窗口中，然后返回工作表界面，在用鼠标右键单击 A 列时将弹出如图 8-10 所示菜单，单击菜单中的任意子菜单可以激活对应的工作表。

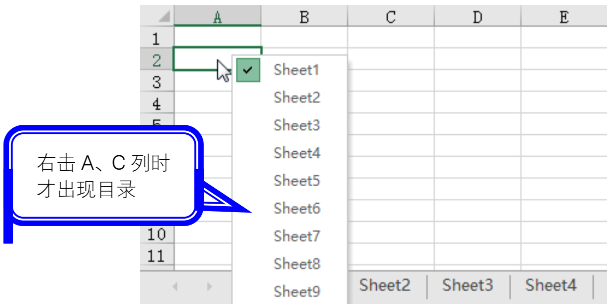


图 8–10　右击 A 列时弹出工作表目录

如果单击 A 列和 C 列以外的任意区域，那么弹出的是默认的快捷菜单，只有右击 A 列或 C 列才会出现“Workbook tabs”菜单，即工作表目录。

【知识补充】

（1）Worksheet_BeforeRightClick 事件是指右击单元格前所触发的事件，也就是说该事件过程中的代码执行完毕后才继续弹出原本的快捷菜单。不过 VBA 提供了关闭原本的快捷菜单的方法，即 Cancel 参数，当赋值为 False 或者采用默认值时，表示继续弹出原本的快捷菜单；赋值为 True，表示关闭原本的快捷菜单。

根据要求，本例代码中限制在右击 A 列或 C 列时调用新的菜单，关闭原本的快捷菜单。

（2） Range("A:A,C:C")表示 A 列和 C 列，不可写作 Range("A:A", "C:C")，它表示从 A 列到 C 列，相当于 A:C 区域。

（3）CommandBars("Workbook tabs")表示 Excel 的工作表目录菜单。

（4）CommandBars.ShowPopup 方法表示将指定的命令栏作为快捷菜单，在指定坐标或当前光标位置显示，其完整语法如下：

```
CommandBars.ShowPopup(x, y)
```

其中 x 和 y 分别表示横坐标和纵坐标，是可选参数。例如以下过程表示在横坐标 400、纵坐标 200 处显示单元格快捷菜单

```
Sub  调用右键菜单()'放置位置：模块中
 Application.CommandBars("cell").ShowPopup   400, 200
End Sub
```

本例案例文件请参考：..\ 第 8 章\ 8-7 在特定区域右击单元格时产生工作表目录.xlsm

8.3.7 事件案例：输入表达式时在右列自动返回计算结果

【案例要求】在 A 列输入表达式时自动转换成计算结果，结果保存在右边一列。

【知识要点】Worksheet_Change 事件。

【程序代码】

```
'工作表 Change 事件（放置位置：“Sheet1”事件代码窗口）
Private Sub Worksheet_Change(ByVal Target As Range)
 On Error Resume Next                                     '有错误时继续执行下一步
 Dim cell As Range                                        '声明变量
 If Target.Columns.Count = 1 And Target.Column = 1 Then   '如果 Target 只有一列，而且列号为 1
For Each cell In Intersect(Target, ActiveSheet.UsedRange) '遍历 Target 区域与已用区域的交集
If Len(cell) > 0 Then                                     '如果单元格中有字符
        cell.Offset(0, 1) = Evaluate(cell.Text)           '在右边一个单元格显示表达式的计算结果
    Else                                                  '否则
        cell.Offset(0, 1) = ""                            '删除右边单元格的值
    End If
  Next cell                                               '转换下一个单元格
  End If
End Sub
```

将以上代码保存在 Sheet1 的代码窗口中，然后返回工作表界面，在 A 列任意单元格中输入表达式，该单元格右边将会出现表达式的计算结果，如图 8-11 所示。

Evaluate 不仅仅可转换包含四则运算的表达式，对于 Excel 自带的任意函数都可以执行计算。在图 8-12 中，A2 的表达式涉及两个函数以及加法运算，转换结果正确无误。

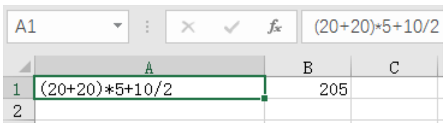


图 8–11　输入表达式时自动转换成值

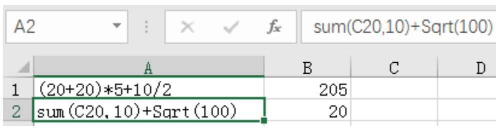


图 8–12　将包括函数的表达式转换成值

如果选择 A 列的多个单元格再录入公式，然后按【Ctrl+Enter】组合键结束，那么事件过程仍然可以正常的工作，并且将会逐一计算 Target 区域的每一个单元格。

【知识补充】

（1）Worksheet_Change 事件表示修改代码所在工作表中的任意单元格时触发的事件，参数 Target 代表当前被修改数据的区域或者单元格。由于 Evaluate 不能批量转换，所以需要配合 For Each...Next 循环语句逐个转换表达式。

（2）Evaluate 函数可以将文本形式的表达式转换成值，将文本形式的对象名称转换成对象，在实际工作中有较高的实用性。不过它的参数有长度限制，只能转换长度在 1 到 256 之间的表达式。

（3）当用户删除 A 列时，不管 A 列有多少数据，For Each...Next 循环都执行 1048576 次运算。为了避免这种情况，本例只在 Intersect(Target, ActiveSheet.UsedRange) 中循环，从而避免不必要的资源浪费，提升代码执行效率。

（4）虽然本例的事件是 Change 事件，而事件中的代码又会修改单元格的值，但是由于事件中限制触发事件的条件是 A 列，而代码的计算结果产生在 B 列，所以不会产生事件的连锁反应。

本例案例文件请参考：..\ 第 8 章\ 8-8 输入表达式时在右列自动返回计算结果.xlsm

8.3.8　事件案例：单击目录时可打开隐藏的工作表

【案例要求】工作簿中包含若干工作表，其中目录工作表中的 B 列有所有工作表的目录，其中部分工作表处于隐藏状态，单击目录链接时无法打开目标工作表。现要求利用 VBA 突破此限制。

【知识要点】Worksheet_FollowHyperlink 事件。

【程序代码】

```
'工作表 FollowHyperlink 事件（放置位置：“目录”事件代码窗口）
Private Sub Worksheet_FollowHyperlink(ByVal Target As Hyperlink)
    Dim linkStr As String               '声明一个变量，用于保存超链接的目标工作表名称
  '提取超链接的目标工作表名称，即链接地址中“!”前面的部分
    linkStr = Mid(Target.SubAddress, 1, InStr(Target.SubAddress, "!") - 1)
    Sheets(linkStr).Visible = True              '将超链接的目标工作表显示出来
    Application.Goto Sheets(linkStr).Range(Target.SubAddress) '选择目标工作表的目标单元格
End Sub
```

将此事件过程的代码保存在目录工作表的代码窗口中，然后返回工作表界面。由于周二工作表处于隐藏状态，所以通过单击 B3 单元格测试事件过程的有效性。单击后发现事件过程的代码有效，可以正常打开目标地址。



图 8–13　工作表目录

【知识补充】

（1）Worksheet_FollowHyperlink 表示在单击超链接时触发的事件，参数 Target 表示触发本事件的单元格。

（2）当工作表处于隐藏状态时，单击超链接不可激活此工作表，所以本例的思路是先从 Target.SubAddress 中提取工作表名称，然后将工作表的 Visible 属性赋值为 True，最后通过 Application.Goto 方法打开超链接的目标地址。

（3）Target.SubAddress 表示单元格的超链接地址，它包含工作表名称和单元格地址，中间用"!"分隔开。本例利用 InStr 函数计算“!”的位置，然后通过 Mid 函数从 Target.SubAddress 中的第 1 位开始提取字符，提取长度为“!”的位置减 1，提取结果刚好等于 Target.SubAddress 中的工作表名称。也可以改用 Evaluate 实现同等功能，因为 Target.SubAddress 是包含工作表与单

元格地址的文本，利用 Evaluate 函数可以将它转换成 Range 对象，然后再利用 Parent.Name 获取它的父对象名称，结果即为目标工作表的名称。完整代码如下：

```
Evaluate(Target.SubAddress).Parent.Name
```

（4）Target.SubAddres 中虽然包含工作表名称和单元格地址，但是 Range(Target.SubAddres) 或者 Evaluate(Target.SubAddress) 都只能获取其中的 Range 对象，前面必须添加父对象 Sheets(linkStr)或者 Range(Target.SubAddress) .Parent 才能准确地引用目标 Range 对象。

本例案例文件请参考：..\ 第 8 章\ 8-9 单击目录时可打开隐藏的工作表.xlsm

8.3.9 事件案例：实时保护已录入数据的单元格

【案例要求】每输入一个单元格就保护一个单元格，禁止修改已经保护的单元格。

【知识要点】Worksheet_Change 事件。

【程序代码】

首先使用【Ctrl+A】组合键全选工作表的所有单元格，然后在“设置单元格格式”对话框的“保护”选项卡中取消勾选“锁定”复选框，最后进入 VBE 界面，在工作表 Sheet1 的代码窗口中录入以下事件过程代码：

```
'工作表 Change 事件（放置位置：“Sheet1”事件代码窗口）
Private Sub Worksheet_Change (ByVal Target As Range)
      If Len(Target(1)) > 0 Then            '如果不是删除数据
     '如果工作表已保护则解除保护
      If ActiveSheet.ProtectContents Then ActiveSheet.Unprotect "123"
        Target.Locked = True                '锁定 Target 区域
        Target.FormulaHidden = False '取消隐藏 Target 区域
        '保护工作表，密码为 123，保护后允许修改锁定单元格之外的一切操作
        ActiveSheet.Protect "123", False, True, False, True, True, True, True, True, True, True, True, True,
True, True, True
    End If
End Sub
```

在工作表 Sheet1 的任意单元格录入数据，当按【Enter】键后，该单元格或者区域就会自动进入锁定状态，禁止修改，但会自动忽略未录入数据的空单元格。

【知识补充】

（1）本例要求输入数据后保护有数据的单元格，所以必须采用 Worksheet_Change 事件，在修改单元格的值后触发此事件。同时为了避免按下【Delete】时会保护空单元格，在代码中采用了条件语句判断 Target(1)的字符长度是否大于 0，只有字符长度大于 0 时才执行锁定工作表的相关代码。

（2）由于 Target 有可能是区域，所以不使用“If Len(Target) > 0 Then”，否则执行代码时会出错。要么只判断 Target 区域的第一个单元格是否空白，要么使用工作表函数 Counta 计算区域中的非空单元格个数，从而判断用户是否按下了【Delete】键。

（3）Worksheet.Protect 保护工作表时只对 Locked 属性值为 True 的单元格生效，而默认状态下的单元格 Locked 属性值则为 True，所以在执行过程前需要将工作表中所有单元格的 Locked 属性设置为 False。

（4）为了在保护锁定单元格时不影响图表、图片、透视表的正常操作，以及避免不能插入行、删除行等问题，需要将 DrawingObjects 和 Scenarios 两个参数赋值为 False，其余参数设置为 True。

本例案例文件请参考：..\ 第 8 章\ 8-10 实时保护已录入数据的单元格.xlsm

8.3.10　事件案例：在状态栏显示当前科目的不及格人数

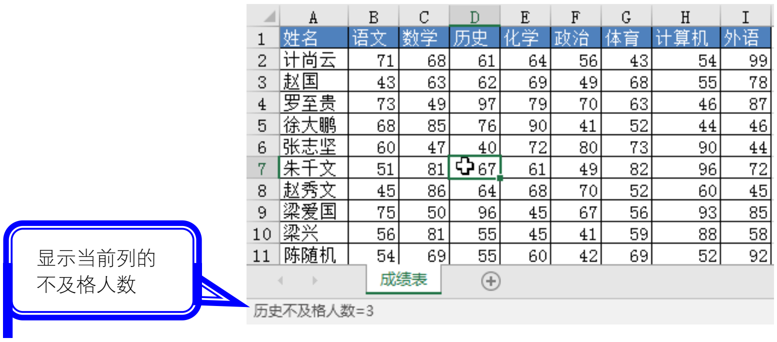
【案例要求】工作表中有若干个科目的成绩，要求选择哪一列就在状态栏显示那一列的科目所对应的不及格人数。

【知识要点】Worksheet_SelectionChange 事件。

【程序代码】

```
'工作表 SelectionChange 事件（'放置位置：“Sheet1”事件代码窗口）
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
    '如果活动单元格所在列大于 1，而且小于等于最后一个非空列
     If Target.Column > 1 And Target.Column <= Cells(1, Columns.Count).End(xlToLeft).Column Then
        '在状态栏显示科目名称，及该科目的不及格人数
        Application.StatusBar = Target.EntireColumn.Cells(1) & "不及格人数=" &
WorksheetFunction.CountIf(Target.EntireColumn, "<60")
    Else '否则
        Application.StatusBar = "" '清除状态栏的自定义信息
    End If
End Sub
```

将以上代码录入到成绩表的代码窗口中，然后在返回工作表界面选择任意单元格，如果所选列处于第 2 列到最后一个非空列之间，那么将在状态栏中显示该列的科目名称与不及格人数，如图 8-14 所示。



	A	B	C	D	E	F	G	H	I
1	姓名	语文	数学	历史	化学	政治	体育	计算机	外语
2	计尚云	71	68	61	64	56	43	54	99
3	赵国	43	63	62	69	49	68	55	78
4	罗至贵	73	49	97	79	70	63	46	87
5	徐大鹏	68	85	76	90	41	52	44	46
6	张志坚	60	47	40	72	80	73	90	44
7	朱千文	51	81	67	61	49	82	96	72
8	赵秀文	45	86	64	68	70	52	60	45
9	梁爱国	75	50	96	45	67	56	93	85
10	梁兴	56	81	55	45	41	59	88	58
11	陈随机	54	69	55	60	42	69	52	92

图 8-14　在状态栏显示“历史”的不及格人数

如果选择其他列，那么状态栏的自定义信息将自动清除。

【知识补充】

（1）由于要求在选择单元格时执行命令，所以本例采用 Worksheet_SelectionChange 事件。

（2）Worksheet_SelectionChange 事件对代码所在工作表中每个单元格都生效，而实际上只有不到 10 列的区域有成绩，所以需要通过 If 条件语句限制过程的作用区域。

（3） Application.StatusBar 表示状态栏，可以将任意非错误值的信息显示在状态栏中，不过状态栏的空间有限，应当尽量将字符控制在 50 字以内，否则可能显示不完整（实际显示字符受显示器大小和屏幕分辨率大小影响）。

如果需要恢复状态栏的显示内容，对 Application.StatusBar 赋值空文本即可。赋值为空文本的作用不是让状态栏什么都不显示，而是按系统原本的显示方式显示自身的内容。如果需要状态栏中什么都不显示，可以对 Application.StatusBar 赋值为空格。

（4）Target.EntireColumn.Cells(1) 表示当前所选区域的首列的第一个单元格，如果科目所在

行为第二行，那么 Cells 的参数需要改为 2。

（5）Countif 是工作表函数，不是 VBA 中的函数，不能在 VBA 中直接使用。在调用工作表函数时要在函数名称前添加“WorksheetFunction.”，VBA 会在录入小圆点后自动弹出所有函数列表，从中选择即可，不需要手动逐个字母录入。

本例案例文件请参考：..\ 第 8 章\ 8-11 在状态栏显示当前科目的不及格人数.xlsm

8.3.11 事件案例：通过数据有效性的下拉列表调用对应的图片

【案例要求】图 8-15 中的 Sheet1 表 A 列是图片名称，B 列是与 A 列同名的图片。在 Sheet2 表中的 A1 单元格通过有效性创建了一个下拉列表，可在列表中随意选择与 sheet1 的 A 列相同的图片名称。现要求在修改 sheet2 中 A1 的值时调用 Sheet1 的同名图片。



图 8–15 图片列表

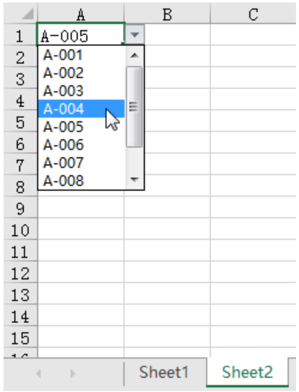


图 8–16 数据有效性的下拉列表

【知识要点】Worksheet_Change 事件。

【程序代码】

```
'工作表 Change 事件（放置位置：“Sheet2”事件代码窗口）
Private Sub Worksheet_Change(ByVal Target As Range)
    '当有错误时执行下一步（A1 的下拉列表没有 TopLeftCell 属性，所以执行 For 循环语句时会出错）
     On Error Resume Next
     Application.ScreenUpdating = False          '关闭屏幕更新
      If Target.Address = "$A$1" Then            '如果当前被修改的单元格的地址是 A1
     '如果本工作表中图形对象个数大于 1 个（A1 的下拉列表也算一个）
      If Me.Shapes.Count > 1 Then
        Dim sh As Shape                          '声明一个图形对象变量
        For Each sh In Me.Shapes                 '遍历所有图形对象
          If sh.TopLeftCell.Address = "$B$1" Then '如果当前对象的左上角单元格是 B1
            sh.Delete                            '删除此对象
            Exit For                             '退出循环语句（没必要继续执行剩余的循环）
          End If
          Next sh
      End If
    Sheet1.Shapes(Range("a1").Value).Copy        '复制 Sheet1 中名字等于本工作表 A1 的图形对象
    ActiveSheet.Paste                            '粘贴到本工作表中
    Selection.Left = Range("b1").Left            '将图形对象的左边距设置为 B1 的左边距
    Selection.Top = Range("b1").Top              '将图形对象的上边距设置为 B1 的上边距
  End If
```

```
    Application.ScreenUpdating = True                    '恢复屏幕更新
End Sub
```

将以上事件过程代码录入到 Sheet2 的代码窗口中，然后返回工作表界面，在 Sheet2 工作表中修改 A1 单元格的值，A2 单元格将自动调用 Sheet1 中同名的图片，如图 8-17 所示。

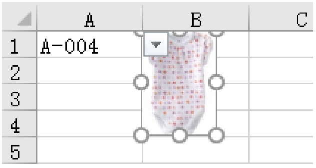


图 8–17　修改 A1 的值时调用同名图片

【知识补充】

（1）由于要求在修改 Sheet2 的 A1 单元格时调用图片，所以本例事件过程的代码必须保存在 Sheet2 的代码窗口中，否则代码不会执行。

（2）通过数据有效性创建的下拉列表属于图形对象，包含在 Shapes 对象集合中，所以 For Each…Next 循环语句遍历图形对象时会调用该下拉列表。同时，由于有效性创建的下拉列表没有 TopLeftCell 属性，那么循环到该对象时代码会出错，所以在过程的首行必须加上“On Error Resume Next”语句防错，否则代码无法执行下去。

（3）在事件过程中使用 Me 时，Me 代表事件过程的主体对象。例如在工作表事件中，Me 代表工作表本身；在工作簿事件中使用 Me 时，Me 代表工作簿；在窗体事件中使用 Me 时，Me 则代表窗体。

（4）Shape.Copy 方法没有参数，只能将图形对象复制到剪贴板中，而 Range.Copy 方法则可以指定粘贴的目标单元格，两者的使用方法不同。所以在复制图形对象时，需要分多步执行：先复制对象，然后粘贴，最后设置左边距和上边距，或者继续设置其高度与宽度。

（5） Shape.TopLeftCell 表示图形对象左上角的单元格，Shape.BottomRightCell 则表示右下角的单元格。

（6）本例的事件过程基于一个前提——Sheet1 中的图片已经命名为 A 列中同行的值，即不再是“图片 1”、“图片 2”这种名称，而是“A-001”、“A-002”。如果插入到 Sheet1 中的图片未经过重命名，那么本例的事件过程无法正常执行。可以通过以下代码批量命名：

```
Sub 批量命名图片()                                          '放置位置：模块中
    On Error Resume Next                                   '有错误时执行下一步
    If Sheet1.Shapes.Count > 0 Then                        '如果 Sheet1 的图形对象数量大于 0
        Dim sh As Shape                                    '声明一个图形对象变量
        For Each sh In Sheet1.Shapes                       '遍历所有图形对象
            sh.Name = sh.TopLeftCell.Offset(0, -1).Value   '命名为左边的单元格的值
        Next sh
    End If
End Sub
```

本例案例文件请参考：..\ 第 8 章\ 8-12 通过数据有效性的下拉列表调用对应的图片.xlsm

8.4　工作簿事件详解

工作簿级事件高于工作表级事件，它包含所有工作表级事件。

Excel 2010 有 36 类工作簿事件，本节提供所有的工作簿事件名称与功能说明，并对其中实用性较强的几个事件提供案例演示。

8.4.1 工作簿事件列表

不同版本的 Excel 有不同数量的工作簿事件，几乎每次升级 OFFICE 都会新增一些事件。在 Excel 2016 中有 40 类工作簿事件，其中包括了 10 多类工作表事件，也有一些工作簿特有的事件。表 8-4 罗列了 Excel 2016 所支持的所有工作簿事件。

表 8-4 Excel 2016 的工作簿事件列表

工作簿事件	说　明
Activate	在激活工作簿时触发此事件
AddinInstall	安装加载宏工作簿触发此事件
AddinUninstall	卸载加载宏工作簿触发此事件
AfterSave	在保存工作簿之后触发此事件
AfterXmlExport	在Excel保存或导出指定工作簿中的XML数据之后触发此事件
AfterXmlImport	在刷新XML数据连接或新的XML数据被导入工作簿后触发此事件
BeforeClose	在关闭工作簿之前触发此事件
BeforePrint	在打印指定工作簿（或者其中的任何内容）之前触发此事件
BeforeSave	在保存工作簿之前触发此事件
BeforeXmlExport	在Excel保存或导出指定工作簿中的XML数据之前触发此事件
BeforeXmlImport	在刷新现有的XML数据连接或新的XML数据被导入工作簿之前触发此事件
Deactivate	在工作簿被停用时触发此事件（也可理解为激活其他工作簿时触发此事件）
ModelChange	在数据模型更改时触发此事件
NewChart	在工作簿中创建新图表时触发此事件
NewSheet	在在工作簿中新建工作表时触发此事件
Open	打开工作簿时触发此事件
PivotTableCloseConnection	在数据透视表的连接关闭之后触发此事件
PivotTableOpenConnection	在数据透视表的连接打开之后触发此事件
RowsetComplete	在OLAP数据透视表上深化记录集或调用行集操作时触发此事件
SheetActivate	在激活工作簿中任意工作表时触发此事件
SheetBeforeDelete	删除任意工作表后触发此事件
SheetBeforeDoubleClick	在双击工作簿中任意单元格时触发此事件，此事件优先于双击操作
SheetBeforeRightClick	在右击工作簿中任意单元格时触发此事件，此事件优先于默认的右击操作，即先执行本事件，然后再决定是否弹出默认的快捷菜单
SheetCalculate	在工作簿中任意公式执行重算时触发此事件
SheetChange	当更改工作簿中任意单元格的值时触发此事件
SheetDeactivate	当任何工作表被停用时触发此事件(也可理解为工作簿中的工作表变成非活动工作表时触发此事件，参数Sh代表执行事件时的活动工作表）
SheetFollowHyperlink	在单击工作簿中任意超链接时触发此事件
SheetLensGalleryRenderComplete	在任意工作表中使用数据分析时触发此事件
SheetPivotTableAfterValueChange	在编辑或重新计算(针对包含公式的单元格)数据透视表中的单元格或单元格区域后触发此事件
SheetPivotTableBeforeAllocateChanges	在向数据透视表应用更改前触发此事件
SheetPivotTableBeforeCommitChanges	在针对OLAP数据源提交对数据透视表的更改前触发此事件
SheetPivotTableBeforeDiscardChanges	在放弃对数据透视表所做的更改之前触发此事件
SheetPivotTableChangeSync	在对数据透视表进行更改之后触发此事件
SheetPivotTableUpdate	在数据透视表的工作表更新之后触发此事件
SheetSelectionChange	在工作簿中任意工作表上的选区发生更改时触发此事件
SheetTableUpdate	连接到数据模型的查询表的工作表上更新后触发此事件
Sync	在作为“文档工作区”一部分的工作簿的本地副本与服务器上的副本进行同步时触发此事件（新版本的Excel已弃用此事件）

续表

工作簿事件	说　明
WindowActivate	在工作簿窗口被激活时触发此事件
WindowDeactivate	任何工作簿窗口被停用时触发此事件
WindowResize	任何工作簿窗口调整大小时触发此事件

其中最常用的事件包括 Activate、AfterSave、BeforeClose、BeforePrint、BeforeSave、NewChart、NewSheet、Open、SheetActivate、SheetBeforeDoubleClick、SheetBeforeRightClick、SheetCalculate、SheetChange、SheetFollowHyperlink 和 SheetSelectionChange 等事件。

8.4.2　事件案例：记录工作簿打开次数

【案例要求】记录工作簿的打开次数。

【知识要点】Workbook_Open 事件。

【程序代码】

```
'工作簿 Open 事件（放置位置：Thisworkbook 代码窗口）
Private Sub Workbook_Open()                          '打开工作簿时执行此事件
  Dim Open_Count As Long                             '声明变量，用于获取自定义的“打开次数”属性
  With ActiveWorkbook.CustomDocumentProperties       '引用活动工作簿的自定义属性
    On Error Resume Next                             '有错误时继续执行下一句
    Open_Count = .Item("打开次数").Value              '获取“打开次数”属性的值
    If Open_Count = 0 Then                           '如果此时变量的值仍然是 0
      '为工作簿添加新的属性“打开次数”，其默认值为 1
      .Add Name:="打开次数", LinkToContent:=False, Type:= msoPropertyTypeNumber, Value:=1
    Else                                             '否则（表示曾经添加了自定义属性，且有赋值）
      .Item("打开次数") = Open_Count + 1              '在原有的属性值基础上累加 1
    End If
    Application.StatusBar = "打开次数：" & Open_Count '在状态栏显示打开次数
  End With
End Sub
```

将以上代码录入到 ThisWorkbook 的代码窗口中，然后保存工作簿并重启，在状态栏即可看到工作簿打开次数的信息。首次打开工作簿时其自定义属性“打开次数”的值为 1，在第三次打开时则状态栏将显示“打开次数：3”。

【知识补充】

（1） Workbook_Open 事件表示打开工作簿时执行的事件，与命名为“Auto_open”的过程的功能一致。不过 Workbook_Open 事件代码必须保存在 ThisWorkbook 中，而“Auto_open”过程代码则只能保存在模块中。

（2） Workbook.CustomDocumentProperties 表示工作簿的自定义属性集合，可以通过参数访问其子对象，也可以通过 CustomDocumentProperties.add 方法创建新的自定义属性。

（3）工作簿自身没有工作簿开启次数的记录，所以本例采用 CustomDocumentProperties. add 方法对工作簿创建一个自定义的属性，并且配合 Workbook_Open 事件让工作簿每打开一次就累加一次开启记录。CustomDocumentProperties.add 方法语法如下：

```
CustomDocumentPropertie.Add(Name, LinkToContent, Type, Value, LinkSource)
```

其中各参数的含义如表 8-5 所示。

表 8-5　CustomDocumentProperties.Add的参数说明

名　　称	必选/可选	说　　明
Name	必选	自定义属性的名称
LinkToContent	必选	用于指定属性是否链接到文档中的内容。如果参数值为True，则必须设置 LinkSource参数，表示属性值的来源，如果值为False，则须要指定Value参数
Type	可选	该属性的数据类型。可以是以下msoPropertyBoolean、msoPropertyDate、msoPropertyFloat、msoPropertyNumber 或 msoPropertyString中的任意一个
Value	可选	为属性指定值。当LinkToContent值为False时必须指定此参数
LinkSource	可选	为属性指定值，当LinkToContent值为False时忽略此参数

本例中自定义属性的名称为“打开次数”，其赋值类型为数值，所以 Type 参数采用 msoPropertyNumber。

（4）由于第一次执行代码时不存在自定义属性“打开次数”，所以执行代码“Open_Count = .Item("打开次数").Value”时会出错，导致赋值不成功，此时变量 Open_Count 的值仍然是默认值 0。如果不是第一次执行代码，那么该代码不会出错，变量 Open_Count 将会被赋值为属性“打开次数”的值。

为了避免再次执行代码时出错，本例在该语句之前插入防错的语句“On Error Resume Next”，从而让代码继续执行下去。

在执行代码“Open_Count = .Item("打开次数").Value”后，如果是首次执行，那么变量 Open_Count 的值仍然是 0，否则变量 Open_Count 将被赋值为实际的文件打开次数。

后面的代码将根据变量 Open_Count 的值来决定处理方式，当变量 Open_Count 的值为 0 时，创建一个名为“打开次数”的属性，且赋值为 1；当变量 Open_Count 的值不等于 0 时，则在属性 Open_Count 的原值上累加 1.

（5）“On Error Resume Next”语句的功能是忽略出错的代码，继续执行下一行的代码。在本书第 9 章中将详细阐述关于处理代码错误的技巧。

本例案例文件请参考：..\ 第 8 章\ 8-13 记录工作簿打开次数.xlsm

8.4.3　事件案例：显示活动工作表中的产量达标率

【案例要求】图 8-18 中包含若干工作表，以月份命名，每个工作表中保存一个月的产量数据。现要求在状态栏显示活动工作表中的产量达标率，切换新工作表时状态栏的值自动更新（产量达到 1000 时就算达标）。

	A	B	C	D	E	F
1	姓名	产量				
2	计尚云	986				
3	赵国	881				
4	罗至贵	1200				
5	徐大鹏	900				
6	张志坚	1139				
7	朱千文	1111				
8	赵秀文	945				
9	梁爱国	1142				
10	梁兴	929				
11	陈随机	919				

一月　二月　三月　四月　五月　六月

图 8–18　产量表

【知识要点】Workbook_SheetActivate 事件。

【程序代码】

```
'工作簿 SheetActivate 事件（放置位置：Thisworkbook 代码窗口）
Private Sub Workbook_SheetActivate(ByVal Sh As Object)
  On Error Resume Next  '出错时继续执行下一步
'利用工作表函数 CountIf 计算活动工作表的 B 列中大于等于 1000 的人数
'再利用工作表函数 Counta 计算 B 列中的数值总个数
'两者相除即得到达标率，最后通过 Format 转换成百分比形式
Application.StatusBar = Sh.Name & "达标率为：" & Format(WorksheetFunction.CountIf(Range("B:B"),
">=1000") / WorksheetFunction.Counta(Range("B:B")), "0.00%")
End Sub
```

将以上代码录入到 Thisworkbook 的代码窗口中，然后返回工作表界面，任意切换工作表，在状态栏中将显示活动工作表的产量达标率，如图 8-19 所示。

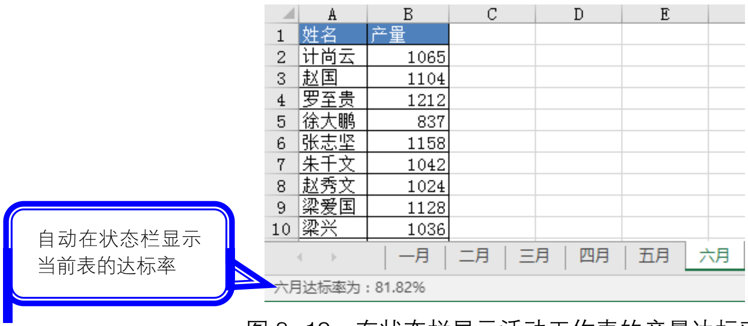


图 8–19　在状态栏显示活动工作表的产量达标率

【知识补充】

（1）Workbook_SheetActivate 事件是激活工作簿中任意工作表时触发的事件，参数 Sh 代表活动工作表。由于 Workbook_SheetActivate 是工作簿级事件，所以只有切换代码所在工作簿中的工作表时才会触发事件，单击其他工作簿中的工作表时不会触发此事件。

（2）调用工作表函数 Countif 时，其第一参数需要使用“Range("B:B")”而不是“B:B”或者“"B:B"”。第二参数则与单元格中书写公式时的用法一致，使用字符串即可。

（3）工作表函数 Counta 用于计算区域中的数值个数，如果区域中不存在数值则返回 0。本例用数值个数作为除数，所以如果 B 列是空列时程序将会出错，为此必须在过程中使用防错语句“On Error Resume Next”，在本书第 9 章会详细讲述防错语句的更多应用。

（4）Workbook_SheetActivate 事件对代码所在工作簿中的任意工作表都生效，所以每切换一次工作表会执行一次本事件过程，状态栏的信息也相应地更新。

本例案例文件请参考：..\ 第 8 章\ 8-13 记录工作簿打开次数.xlsm

8.4.4　事件案例：保存工作簿时备份文件

【案例要求】保存工作簿时自动备份工作簿，备份的文件采用原工作簿的文件名称，只需要在后面加上“(Bak)”作为标识即可，其文件内容与文件格式不变。

【知识要点】Workbook_AfterSave 事件。

【程序代码】

```
'工作簿 AfterSave 事件（放置位置：Thisworkbook 代码窗口）
Private Sub Workbook_AfterSave(ByVal Success As Boolean)  '保存文件后触发此事件
  If Success = True Then      '如果保存成功
```

```
        Dim 后缀名 As String    '声明变量
        '提取文件的后缀名
       后缀名 = Mid(ThisWorkbook.Name, InStrRev(ThisWorkbook.Name, "."), 5)
        '将在原文件名中插入"(Bak)"作为备份文件的名称
        ThisWorkbook.SaveCopyAs Replace(ThisWorkbook.FullName, 后缀名, "(Bak)" & 后缀名)
    End If
End Sub
```

将以上代码录入到 Thisworkbook 的代码窗口中，然后保存文件，如果保存成功，那么在同路径下将自动产生一个备份文件，其文件名取自原文件名称加“(Bak)”，如图 8-20 所示。

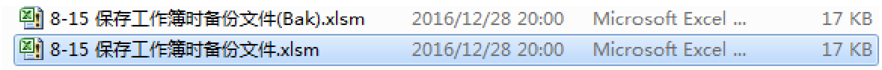


图 8–20　原文件与备份的文件

【知识补充】

（1）Workbook_AfterSave 事件是保存文件后触发的事件，参数 Success 是一个布尔值，如果保存成功则返回 True，保存不成功则返回 False。

（2）由于保存文件时格式不确定，后缀名的长度也不确定，所以不能直接使用最后三位或者四位作为文件的后缀名，必须使用函数计算。

（3）从文件的全名 ThisWorkbook.Name 中提取后缀名用 Mid(ThisWorkbook.Name, InStrRev(ThisWorkbook.Name, "."), 5)，其中 InStrRev 用于计算小数点的位置，Mid 函数则根据位置提取后缀名。

（4）Excel 2016 还有一个 Workbook_BeforeSave 事件，它是在保存文件之前触发的事件。

本例案例文件请参考：..\ 第 8 章\ 8-15 保存工作簿时备份文件.xlsm

8.4.4　事件案例：打印数据前检查资料是否填写完整

【案例要求】活动工作表中 C2、D10:D15、F5 等单元格是必填项目，如果这些单元格忘记填写则禁止打印。

【知识要点】Workbook_BeforePrint 事件。

【程序代码】

```
'工作簿 BeforePrint 事件（放置位置：Thisworkbook 代码窗口）
Private Sub Workbook_BeforePrint(Cancel As Boolean)    '打印前执行此事件过程
    Dim Bl As Boolean                                   '声明一个变量
   '将必填区域的非空单元格个数与必填区域的单元格个数执行比较，结果赋值给变量 Bl
     Bl = (WorksheetFunction.CountA(Range("C2,D10:D15,F5")) <>Range("C2,D10:D15,F5").Cells.Count)
    '如果 Bl 的值是 True 则提示用户
    If Bl Then MsgBox "资料填写不完整，禁止打印", vbOKOnly, "友情提示"
    Cancel = Bl                                         '由 Bl 的值决定是否继续打印
End Sub
```

将以上代码录入到 Thisworkbook 的代码窗口中，然后返回工作表界面打印文件，如果必填区域中有任何单元格属于空白状态，那么将弹出如图 8-21 所示的提示框，并且禁止打印；如果必填区域中没有任何空白单元格，那么可以正常打印。

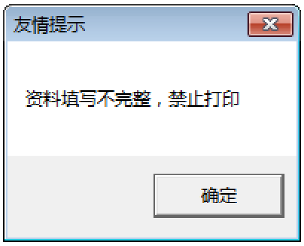


图 8-21　资料不完整时的提示

【知识补充】

（1）Workbook_BeforePrint 事件是打印文件前触发的事件，参数 Cancel 表示是否取消打印，赋值为 True 时表示取消打印，否则允许打印。

（2）检查必填区域 C2、D10:D15、F5 是否存在空单元格时可以使用循环语句逐个判断，也可以用工作表函数 Counta 计算非空单元格个数与区域中的单元格个数是否相等，显然采用函数计算更简洁，所以本例没有使用 For Next 循环语句，不过读者可以用循环语句改写本例代码，练习循环语句的用法。

（3）借用本例的思路也可以实现按时间决定是否允许打印，即只能在某个时间段打印。

本例案例文件请参考：..\ 第 8 章\8-16 打印数据前检查资料是否填写完整.xlsm

8.4.5　事件案例：保存工作簿时更新工作表目录

【案例要求】在模块中已经有一个过程“创建工作表目录”用于创建工作表目录，由于担心删除工作表或者新增工作表后忘记更新目录，所以要求保存文件后自动更新目录。

【知识要点】Workbook_AfterSave 事件。

【程序代码】

```
'工作簿 AfterSave 事件（放置位置：Thisworkbook 代码窗口）
Private Sub Workbook_AfterSave(ByVal Success As Boolean)   '保存文件之后触发事件
    If Success Then                                        '如果保存成功
        Call 创建工作表目录                                 '更新目录
        Application.EnableEvents = False                   '禁止响应事件
        Me.Save                                            '保存代码所在工作簿
        Application.EnableEvents = True                    '恢复响应事件
    End If
End Sub
```

将以上代码录入到 Thisworkbook 的代码窗口中，然后保存文件，Workbook_AfterSave 事件过程会调用模块中的过程“创建工作表目录”，从而更新“工作表目录”中的目录。

如果新建或者删除了原来的工作表，那么本例的事件过程可以自动更新目录，不需要工作表每增减一次就手动执行一次过程“创建工作表目录”。

【知识补充】

（1）Workbook_AfterSave 事件是保存工作簿后触发的事件，参数 Success 表示是否保存成功。如果保存成功则返回 True，否则返回 False。

（2）代码“Me.Save”表示保存代码所在工作簿，Me 等同于 Thisworkbook。

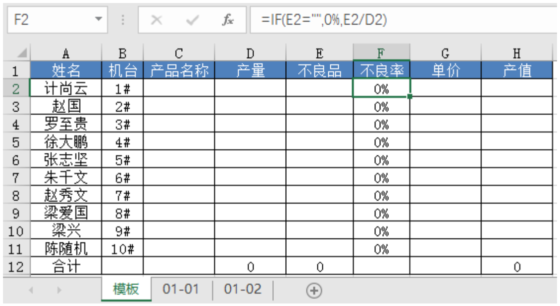
由于在保存工作簿时会再次触发 Workbook_AfterSave 事件，所以在“Me.Save”之前需要禁止响应事件。

本例案例文件请参考：..\ 第 8 章\ 8-17 保存工作簿时更新工作表目录.xlsm

8.4.6 事件案例：新建工作表时调用模板格式

【案例要求】图 8-22 是一个关于每日产量报表的工作簿，每天需要录入昨天的生产数据，即每天创建一个产量表，以昨天的日期命名，在表中录入昨天的生产数据。

为了统一格式及提升操作速度，在工作簿中创建了一个名为“模板”的工作表，表中有固定的格式及公式，只需要输入产品名称、产量和不良品数量，其他所有内容就会自动产生。现要求单击“新建”按钮或者使用【Shift+F11】组合键新建工作表时可以自动调用“模板”工作表的内容，而且以昨天的日期对新表命名，从而避免手动复制文件及命名。



	A	B	C	D	E	F	G	H
1	姓名	机台	产品名称	产量	不良品	不良率	单价	产值
2	计尚云	1#				0%		
3	赵国	2#				0%		
4	罗至贵	3#				0%		
5	徐大鹏	4#				0%		
6	张志坚	5#				0%		
7	朱千文	6#				0%		
8	赵秀文	7#				0%		
9	梁爱国	8#				0%		
10	梁兴	9#				0%		
11	陈随机	10#				0%		
12	合计			0	0			0

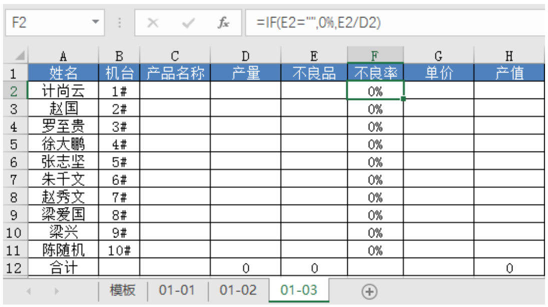
图 8–22 模板工作表

【知识要点】Workbook_NewSheet 事件。

【程序代码】

```
'工作簿 NewSheet 事件（放置位置：Thisworkbook 代码窗口）
Private Sub Workbook_NewSheet(ByVal Sh As Object)          '新建工作表时触发事件
  Application.DisplayAlerts = False                        '关闭提示
  Sh.Delete                                                '删除新建的工作表
'将“模板”工作表复制到最后一个工作表的右边
Worksheets("模板").Copy after:=Sheets(Sheets.Count)
'有错误时继续执行（当已经存在以昨日日期命名的表时执行下面一行代码就会出错）
  On Error Resume Next
  Sheets(Sheets.Count).Name = Format(Date - 1, "mm-dd")    '用昨日日期对复制的模板重命名
End Sub
```

将以上代码复制到 Thisworkbook 代码窗口中，然后返回工作表界面。假设今日是 1 月 4 日，那么使用【Shift+F11】组合键新建工作表时会产生图 8-23 所示结果。在图中新建的工作表不是一个空白的 Sheet1，而是与“模板”内容一致的名为“01-03”的工作表。



	A	B	C	D	E	F	G	H
1	姓名	机台	产品名称	产量	不良品	不良率	单价	产值
2	计尚云	1#				0%		
3	赵国	2#				0%		
4	罗至贵	3#				0%		
5	徐大鹏	4#				0%		
6	张志坚	5#				0%		
7	朱千文	6#				0%		
8	赵秀文	7#				0%		
9	梁爱国	8#				0%		
10	梁兴	9#				0%		
11	陈随机	10#				0%		
12	合计			0	0			0

图 8–23 新建工作表时自动调用模块

【知识补充】

（1）Workbook_NewSheet 事件是指新建工作表时触发的事件，参数 Sh 代表新建的工作表。

（2）由于防错语句在本书第 9 章才会讲到，因此本案例中并没有防错处理，当多次按下【Shift+F11】组合键时会新建多个工作表，而更理想的方式是配合防错语句使用，避免每天新建多个工作表，因为本工作簿仅用于存放日报表，每一天仅需要创建一个新工作表。

（3）本例调用模板的思路是删除新建的工作表，然后复制模板工作表并改名。也可以采用新的思路，例如复制模板中的数据和格式到新建的工作表中，然后对新表命名，代码如下：

```
Private Sub Workbook_NewSheet(ByVal Sh As Object)    '新建工作表时触发事件
  Worksheets("模板").Cells.Copy Sh.Range("a1")       '将“模板”工作表的数据复制到新建的工作表中
'有错误时继续执行（当已经存在昨日日期命名的表时执行下面一行代码时就会出错）
  On Error Resume Next
 sh.Name = Format(Date - 1, "mm-dd")                 '用昨日日期对复制的模板文件重命名
End Sub
```

（4）Date 表示当前系统日期，其实是一个数值，所以如果需要取得昨天的日期，那么直接减 1 即可。

（5）在其他工作簿中新建工作表不会调用工作簿中的事件过程，因此也不会调用模板。

本例案例文件请参考：..\ 第 8 章\ 8-18 新建工作表时调用模板格式.xlsm

8.4.7　事件案例：禁止修改总表名称

【案例要求】工作簿中有 6 个工作表，其中第 6 个工作表为总表。6 个工作表的排列方式如图 8-24 所示，而它们的工作表名称与代码名称的对应关系则如图 8-25 所示。



图 8–24　工作表布局方式



图 8–25　工作表的代码名称

现要求通过代码禁止修改“总表”的名称，从而确保汇总时不会出错。

【知识要点】Workbook_SheetDeactivate 事件和 Workbook_SheetSelectionChange 事件。

【程序代码】

```
'工作簿 SheetDeactivate 事件（放置位置：Thisworkbook 代码窗口）
Private Sub Workbook_SheetDeactivate(ByVal Sh As Object)
    Sheet6.Name = "总表"                '对 Sheet6 表重命名
End Sub
Private Sub Workbook_SheetSelectionChange(ByVal Sh As Object, ByVal Target As Range)
    If Sh.CodeName = "Sheet6" Then    '如果 Sh 对象的代码名称是 Sheet6
        Sh.Name = "总表"                '将它重命名为“总表”
    End If
End Sub
```

将以上代码录入到 Thisworkbook 的代码窗口中，然后返回工作表界面修改工作表“总表”的名称，当录入新名称并按下【Enter】键后，原来的“总表”并不会马上恢复为“总表”，仍然

显示修改前的名称，只有切换工作表或者原来的“总表”中的选区变化时才会自动恢复名称“总表”。

【知识补充】

（1）Workbook_SheetDeactivate 事件是活动工作表变成非活动工作表时触发的事件，参数 Sh 代表触发此事件前的活动工作表。

当修改工作表名称后不按【Enter】键，而是单击其他工作表名称时，此事件过程可以及时地恢复“总表”名称。

（2）Workbook_SheetSelectionChange 事件是代码所在工作簿中任意工作表的选区变化时触发的事件。本例的事件过程代码中通过 If 条件语句限定在代码名称为“Sheet6”的工作表才可以执行工作表命名的命令，如果不限制工作表名称，那么在工作簿的任意工作表中更改选区时都会执行命令，从而浪费内存资源。

（3）Worksheet.CodeName 属性表示工作表的代码名称，即 VBE 界面中工程资源管理器中的对象名称，图 8-25 中的 Sheet1、Sheet2 即为工作表的代码名称，而括号中的 A 组、B 组等是工作表名称，可在工作表界面中手动修改。修改工作表名称时工作表的代码名称不会变化，所以不管工作表名称如何修改，总能通过代码名称正确地引用对象。

本例案例文件请参考：..\ 第 8 章\ 8-19 禁止修改总表名称.xlsm

8.4.8 事件案例：新建图表时自动设置为阴影、圆角

【案例要求】新建图表时自动对图表添加阴影，以及让其边框显示为圆角，从而避免每次手动调整图表属性。

【知识要点】Workbook_NewChart 事件。

【程序代码】

```
'工作簿 NewChart 事件（放置位置：Thisworkbook 代码窗口）
Private Sub Workbook_NewChart (ByVal Ch As Chart)      '新建图表时触发事件
    '对 Ch 的父对象的形状区域设置阴影，阴影样式为 msoShadow21
    Ch.Parent.ShapeRange.Shadow.Type = msoShadow21
    Ch.Parent.RoundedCorners = True                   '对 Ch 的父对象设置圆角
End Sub
```

将以上代码录入到 Thisworkbook 的代码窗口中，然后返回工作表界面。选择 A1:C8 区域后打开功能区中的“插入”选项卡→选择“折线图”命令，从而对 A1:C8 区域创建二维折线图表，此时图表会自动产生阴影、圆角，如图 8-26 所示。

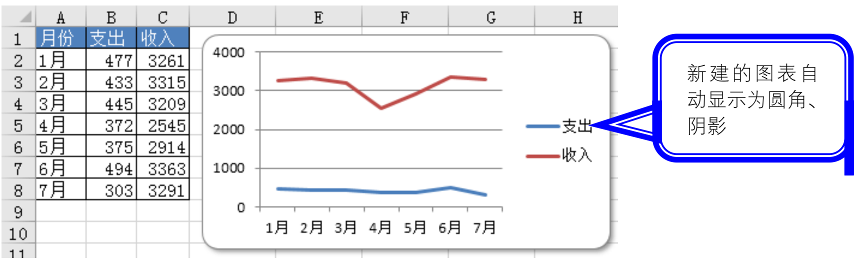


图 8–26 新建图表时自动应用阴影与圆角

【知识补充】

（1）Workbook_NewChart 事件是指工作簿中创建新图表时触发的事件，包括插入和粘贴图

表，修改图表的大小、位置时不会触发事件。参数 Ch 代表新建的图表，它是一个 Chart 对象，它的父对象是 ChartObject，可以对父对象设置圆角属性以及对父对象的形状区域设置阴影。

（2）Chart 对象没有 Shadow 属性，Chart 的父对象也没有 Shadow 属性，记住这一点很重要。

（3）可以直接设置阴影 Shadow 的颜色、偏移量等，但是直接调用内置的阴影样式更简单。

对 Shadow.Type 赋值即可设置阴影样式，Excel 内部提供几十种样式可选，可以通过录制宏取得每个内容阴影样式的编号。

本例案例文件请参考：..\ 第 8 章\ 8-20 新建图表时自动设置为阴影、圆角.xlsm

8.5 应用程序级事件详解

应用程序级事件包含了所有工作表事件和工作簿事件，另外还有应用程序级别的专用事件，所以应用程序级事件很多，使用起来远比工作表事件和工作簿事件复杂。

不过实际工作中极少使用应用程序级事件，本章仅举两例，展示应用程序级事件的调用思路。

8.5.1 应用程序与类

Excel 2016 中包括 47 种应用程序级事件，由于内容较多，此处不再一一罗列出来，读者可以在随书赠送的案例文件中找到，文件夹为“附录”，文件名为“4.应用程序事件.pdf”。

创建应用程序级事件有两种方法，一种需要用到类模块的知识，比较复杂；另一种不需要用类模块知识。本书采用后者演示应用程序级事件。

8.5.2 事件案例：打开任意工作簿时创建工作表目录

【案例要求】通过 Workbook_Open 事件可以实现打开代码所在工作簿时自动创建该工作簿中的工作表目录，现要求将此过程进行扩展，使用户打开任意工作簿时都可以创建工作表目录，目录保存在打开的工作簿的“目录”工作表中。如果不存在“目录”工作表则新建一个工作表，并命名为“目录”。

【知识要点】应用程序 WorkbookOpen 事件。

【程序代码】

双击工程资源管理器中的 ThisWorkbook，在右边的代码窗口中录入以下代码：

```
'放置位置：ThisWorkbook 事件代码窗口中
'声明一个代表 Excel 应用程序的可以触发事件的变量，其中 WithEvents 的功能是让对象变量
'支持事件
Public WithEvents app As Application
'当打开本工作簿时执行本事件过程，这是一个工作簿事件
Private Sub Workbook_Open()
    Set app = Excel.Application                 '当 Excel 应用程序赋值给变量 app
End Sub
'打开任意工作簿时执行本事件过程，这是一个应用程序事件。参数 Wb 代表刚刚打开的这个工作簿
Private Sub app_WorkbookOpen(ByVal Wb As Workbook)
  Application.ScreenUpdating = False            '关闭屏幕更新，提升代码执行效率
  On Error Resume Next   '有错误时继续下一步（当没有"工作表目录"时删除工作表目录会出错）
  Application.DisplayAlerts = False             '关闭提示（删除非空工作表时会弹出提示框）
```

```
    Wb.Sheets("工作表目录").Delete                '删除 wb 工作簿中已有的工作表目录（假设有的话）
  '在 Wb 工作簿最前面创建一个新工作表，并命名为"工作表目录"
    Wb.Sheets.Add(before:=Sheets(1)).Name = "工作表目录"
    Range("A1") = "编号"                          '写入 A 列标题
    Range("B1") = "目录"                          '写入 B 列标题
    Dim i As Integer                              '声明一个变量，用于遍历 Wb 工作簿中的工作表
    For i = 2 To Wb.Sheets.Count                  '遍历 WB 工作簿中“工作表目录”以外的所有表
      Cells(i, 1).Value = i - 1                   '在第一列 i 行单元格中写入序号
      '在第 2 列 i 行的单元格中创建带链接的工作表目录
      ActiveSheet.Hyperlinks.Add Anchor:=Cells(i, 2), Address:="", SubAddress:="'" & Wb.Sheets(i).Name &
"'" & "!A1", TextToDisplay:=Wb.Sheets(i). Name, ScreenTip:="单击打开：" & Wb.Sheets(i).Name
    Next
    Range("A2").Select                            '选择 A2 单元格
    ActiveWindow.FreezePanes = True               '冻结窗格（让首行固定，方便查看）
    Application.ScreenUpdating = True             '恢复屏幕更新
  End Sub
```

保存本工作簿，然后重新打开此工作簿，在工作簿中生成一个“工作表目录”，效果如图 8-27 所示。

此时再打开另一个文件“生产表.xlsm”，此文件中也会瞬间生成“工作表目录”，效果如图 8-28 所示。

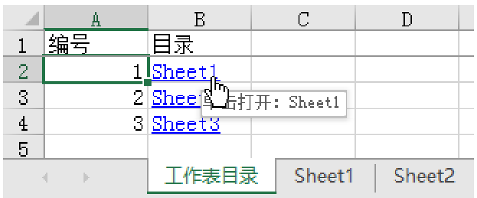


图 8–27　开启代码所在工作簿时自动创建目录



图 8–28　打开其他工作簿时创建目录

事件上，只要 app_WorkbookOpen 事件的代码所在工作簿处于打开状态，那么打开任意工作簿都会调用 app_WorkbookOpen 事件，从而在该工作簿中创建工作表目录，这是应用程序级事件的优势所在。

【知识补充】

（1）应用程序级事件在应用上比工作表级事件和工作簿级事件稍微复杂，操作步骤为如下。

① 在 ThisWorkbook 窗口录入以下代码：

```
  Public WithEvents app As Application
  Private Sub Workbook_Open()
      Set app = Excel.Application
  End Sub
```

② 在代码窗口顶端的对象列表中就可以看到类和应用程序级对象的变量名称，如图 8-29 所示。

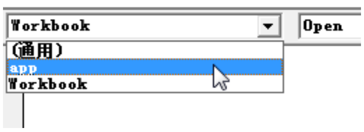


图 8–29　类与应用程序级对象

如果从对象列表中选择“myApp”，那么在右边的过程列表中就能看到应用程序级事件过程名称列表，如图 8-30 所示。

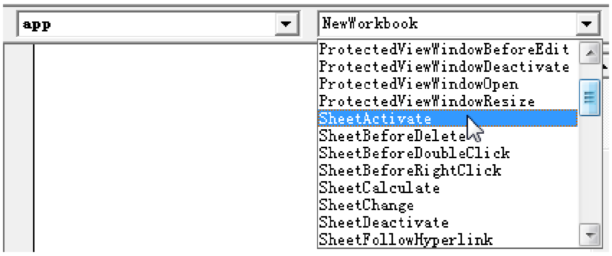


图 8–30　应用程序级事件列表

（2）代码“Public WithEvents app As Application”的功能是创建一个拥有事件的对象变量，但是实际上要让该对象变量的事件生效还需要对变量 app 赋值，因此 Workbook_Open 事件仍然是必要的，需要在此事件中使用“Set app = Excel.Application”来启用。

（3）另外，app_WorkbookOpen 事件是应用程序事件，在打开任意工作簿时触发事件。参数 Wb 代表被打开的工作簿。

如果 app_WorkbookOpen 事件所在工作簿被关闭了，那么此时打开任意工作簿都无法再启用应用程序事件。

（4）通常应用程序级事件都在加载宏中使用。

本例案例文件请参考：..\ 第 8 章\ 8-21 打开任意工作簿时创建工作表目录.xlsm

8.5.3　事件案例：新建工作簿时自动保存

【案例要求】在工作中有时会因为断电或者死机造成新建的文件未保存，可能几小时的工作成果就此丢失。为了避免这种意外造成的损失，现要求新建工作簿时自动弹出保存对话框，从而起到提示、预防作用。

【知识要点】App_NewWorkbook 事件。

【程序代码】

双击工程资源管理器中的 ThisWorkbook，在右边的代码窗口中录入以下代码：

```
'声明一个代表 Excel 应用程序的可以触发事件的变量，其中 WithEvents 的功能是让对象变量
'支持事件
Public WithEvents app As Application
'当打开本工作簿时执行本事件过程，这是一个工作簿事件
Private Sub Workbook_Open()
    Set app = Excel.Application '当 Excel 应用程序赋值给变量 app
End Sub
'新建工作簿时执行本事件过程，这是一个应用程序事件。参数 Wb 代表刚刚新建的这个工作簿
Private Sub app_NewWorkbook(ByVal Wb As Workbook)
   Application.SendKeys "^s"     '发送【Ctrl+S】组合键
End Sub
```

当录入所有代码后，保存并重启此工作簿，然后使用【Ctrl+S】组合键新建一个工作簿，此时可以发现程序会自动弹出“另存为”对话框，表示 App_NewWorkbook 事件已在正常工作。

当然也可以不用重启工作簿，手动执行 ThisWorkbook 中的 Workbook_Open 事件也能达到同等效果，目的都是让 app 对象变量启用事件。

【知识补充】

（1）app_NewWorkbook 事件表示新建工作簿时触发的应用程序事件，参数 Wb 表示新建的工作簿。

（2）Application.SendKeys 方法表示发送快捷键，本例中表示调用【Ctrl+S】组合键打开“另存为”对话框。Application.SendKeys 方法的基本语法如下：

```
Application.SendKeys(Keys, Wait)
```

其中参数 Keys 表示需要发送的组合键，例如【Ctrl+S】、【Ctrl+A】、【Alt+F11】和【Ctrl+N】等组合键。为了避免产生歧义，VBA 代码中不会以原来的形式体现出来，而是采用三个特殊的符号替代常用的三个功能键，如表 8-6 所示。

表 8-6　三个功能键的VBA符号说明

功能键	符　号
Shift	+
Ctrl	^
Alt	%

根据表 8-6 可以得出 VBA 代码中的“^s”即代表【Ctrl+S】组合键，表示另存工作簿；而代码“^+5”则代表【Ctrl+Shift+5】组合键，表示将单元格的数字格式设置为百分比格式。

除 Shift、Ctrl、Alt 三个功能键外，键盘上的其余按键如表 8-7 所示。

表 8-7　键盘按键与VBA代码

按　键	代　码	按　键	代　码
BACKSPACE	{BACKSPACE} 或 {BS}	Ins	{INSERT}
Break	{BREAK}	向左键	{LEFT}
Caps Lock	{CAPSLOCK}	Num Lock	{NUMLOCK}
Clear	{CLEAR}	PageDown	{PGDN}
Delete 或 Del	{DELETE} 或 {DEL}	PageUp	{PGUP}
向下键	{DOWN}	Return	{RETURN}
End	{END}	向右键	{RIGHT}
Enter（数字小键盘）	{ENTER}	Scroll Lock	{SCROLLLOCK}
Enter	~（波形符）	Tab	{TAB}
Esc	{ESCAPE} 或 {ESC}	向上键	{UP}
Help	{HELP}	F1 到 F15	{F1} 到 {F15}
Home	{HOME}		

根据表 8-6 和表 8-7 可知，若想设置打开 VBE 界面的快捷键【Alt+F11】，可采用以下代码：

```
Application.SendKeys "%{F11}"
```

（3）当未保存文件时断电，断电前编辑的任何内容都不可能找回来，而将工作簿保存后再断电，由于 Excel 会自动隔几分钟保存一次，所以重启电脑后仍然可以在缓存文件中找回数据。因此制表过程中要养成先保存文件后编辑内容的习惯。

本例案例文件请参考：..\ 第 8 章\ 8-22 新建工作簿时自动保存.xlsm

8.6　按时间执行代码

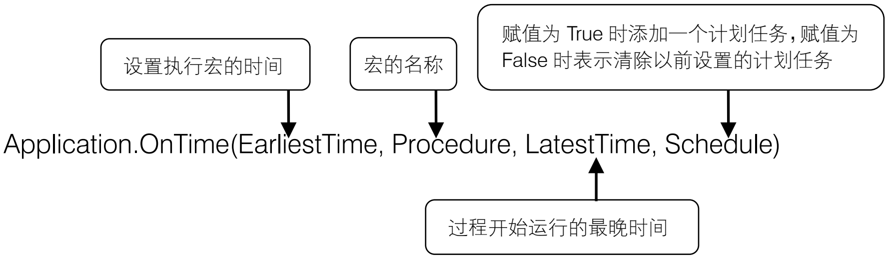
按时间执行代码不需要调用 Excel 的任意事件，不过它与事件有相似之处，下面介绍按时执

行代码的思路。

8.6.1　OnKey 方法的语法分析

使用 Excel 事件可以让代码在满足某条件时自动执行，而使用 Application.OnKey 方法可以让一个或者多个过程在指定的时间自动执行，即创建计划任务。

设置按时执行代码的方法是调用 Application.OnTime，其语法如下：



其中前两个参数是必选参数，后两个参数是可选参数。设置一个计划任务时，通常使用前两个参数即可。

8.6.2　创建计划任务

设置计划任务有两种形式，一是让任务在指定时刻执行，二是让计划任务在某个指定时间段之后执行，例如 5 分钟后或者 10 分钟之后。

假设有一个名为“问候”的过程，需要在 10 点钟执行，那么可使用以下过程创建计划任务：

```
Sub 定时执行 1()                                                     '放置位置：模块中
    Application.OnTime TimeValue("10:00:00"), "问候"                 '10 点钟执行过程“问候”
End Sub
Sub 问候()
    MsgBox "How are you"
End Sub
```

执行过程“定时执行 1”后，到了上午 10 点钟将会自动执行过程“问候”。不过在 10 点钟之前不能关闭 Excel，否则此计划任务自动取消。

假设有一个名为“问候”的过程，需要在 5 秒钟之后执行过间，那么可使用以下过程创建计划任务：

```
Sub 定时执行 2()                                                     '放置位置：模块中
    Application.OnTime Now + TimeValue("0:00:05"), "问候"            '5 秒钟后执行过程“问候”
End Sub
Sub 问候()
    MsgBox "How are you"
End Sub
```

执行过程“定时执行 2”后，5 秒钟将会自动执行过程“问候”。

如果配合递归可以实现每秒钟更新程序。例如在 A1 单元格显示时间，每秒钟更新结果，那么可以采用以下代码：

```
Sub 在 A1 单元格显示时间()                                           '放置位置：模块中
'在 A1 单元格中显示当前时间，显示格式为“hh:mm:ss”
    Range("a1") = Format(Now, "hh:mm:ss")
    Application.OnTime Now + TimeValue("0:00:01"), "在 A1 单元格显示时间"    '每秒钟执行一次
End Sub
```

此过程第一句代码表示在 A1 单元格显示当前的系统时间，同时显示时、分、秒，然后通过 Application.OnTime 方法让过程“在 A1 单元格显示时间”每秒钟执行一次，从而自动更新 A1 的值，达到时钟的效果。

本例案例文件请参考：..\ 第 8 章\ 8-23 创建计划任务.xlsm

8.7 课后思考

1．用什么办法区分事件过程是工作表级事件、工作簿级事件还是应用程序级事件?
2．用什么办法防止在事件过程中产生递归现象?
3．用什么办法确保在录入事件过程的外壳时不至于出错?
4．编写事件过程，在 A1 单元格中记录当前表的打印次数。
5．分析在什么情况下用事件过程，什么情况下用普通的 Sub 过程。

第 9 章　处理代码错误

代码在执行过程中出错是极为常见的现象，如何预防错误是每个程序员必须掌握的技巧。

事实上，也可以将代码错误转换成正能量，通过代码错误来执行某些判断，达成实际的工作需求。本章将具体分析出现代码错误的最常见的原因，以及利用 VBA 内置的错误处理语句获得需要的信息，从而更好地为工作服务。

本章要点

- ◆ 代码错误类型分析
- ◆ 错误处理语句
- ◆ 案例应用

9.1　代码错误类型分析

如果要细化，执行代码过程中产生错误的原因有上百种，不过可以将其中最常见的原因综合为 4 类，本节对其一一进行分析。

9.1.1　版本问题

Excel 2003、Excel 2007、Excel 2010、Excel 2013 和 Excel 2016 等版本都有一些各自独有的对象或者方法、属性，所以不可能确保每句代码都通用于五个版本。

例如 Application.FileSearch 只能在 Excel 2003 中使用，而不能在其他 4 个版本中使用，而 CommandBars.ExecuteMso 方法只能在 Excel 2007 及上版本中才能使用……

从 Excel 2013 开始取消了本地帮助，因此调用帮助的代码在 Excel 2013 和 Excel 2016 中都不能用，只能在 Excel 2007、Excel 2010 中使用。

更多关于版本的问题请参阅本书 4.7.3 节。

9.1.2　参数赋值不当

在 VBA 中，部分函数和方法有多个参数，参数越多编写代码时出错的机率也越大。

在参数赋值方面的错误主要包括 4 个方面：参数不可选、输入太多或者太少参数、赋值的类型错误、未正确使用括号。

1．参数不可选

参数分为可选参数和必选参数。可选参数具有默认值，使用中未给参数赋值时会自动调用默认值；但是必选参数没有默认值，必须对其正确地赋值才可以执行代码。

例如 Range.AutoFill 方法用于填充单元格，有一个必选参数和一个可选参数。第一参数 Destination 表示要填充的单元格，必须对此参数赋值；第二参数 Type 表示填充类型，默认值是 xlFillCopy。这表明在使用 Range.AutoFill 方法时第一参数不可省略，如果填充类型是 xlFillCopy

时可以省略第二参数。

执行以下代码将会出错，提示“参数不可选”，表明未对必选参数赋值：

```
Range("B2").AutoFill
```

解决这类问题最好的办法是查看帮助，在帮助中有每个参数是可选参数还是必选参数的说明。也可以看参数提示，有方括号的参数是可选参数，在图 9-1 中 Type 函数前后有方括号，表明它是可选参数。

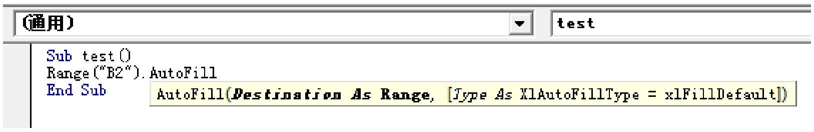


图 9–1　从提示信息看参数是可选参数还是必选参数

2．输入太多或者太少参数

保护工作表的 Worksheet.Protect 方法有 16 个参数，全都是可选参数。所以如果按位置对 16 个参数赋值，那么需要使用 15 个逗号；如果其中部分参数只需要调用默认值，那么使用逗号占位即可。但是在录入过程中有可能录入太多逗号或者太少逗号，从而使得语句无法运行。

解决这种问题其实很简单，VBA 已经提供了相应的参数提示，提示中包含所有的参数名称，还采用粗体字标示当前正在录入的参数。图 9-2 中已经录入了第一参数 Password，正在对 DrawingObjects 参数赋值，所以参数提示信息中的“DrawingObjects”采用粗体字标示。

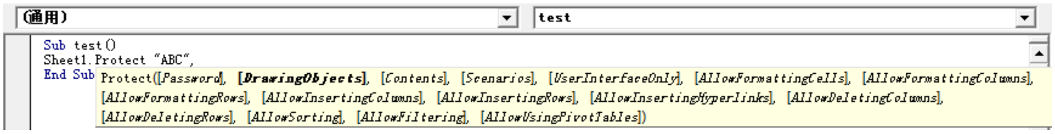


图 9–2　VBA 的参数提示

在录入参数时应尽量查看提示信息，以确保对参数赋值时不会错位。

3．赋值的类型错误

Worksheet.Protect 方法的 16 个参数都是变体型，但是绝大多数方法或者函数的多个参数分别使用不同数据类型的值，只有对参数赋予正确的数据类型的值后，代码才可能正常执行。

例如 Range.AutoFill 方法的第一参数 Destination 是 Range 型，那么只能对参数赋值为单元格或者区域，使用其他任意类型的值都会导致运行代码出错；第二参数 Type 是 XlAutoFillType 型，搜索关键字“XlAutoFillType”即可看到关于此类型的数据的描述为“根据源区域的内容，指定目标区域的填充方式”，其可选项包括 xlFillCopy、xlFillDays、xlFillDefault、xlFillFormats、xlFillMonths、xlFillSeries、xlFillValues、xlFillWeekdays、xlFillYears、xlGrowthTrend 和 xlLinearTrend。所以根据提示和帮助足以确保在对参数赋值时的正确性。

在对参数赋值时，除默认的变体型 Variant 外，VBA 会提示每个参数所允许的数据类型。例如在如图 9-3 所示的参数提示中就表明了 Range.AutoFill 方法的两个参数的数据类型。

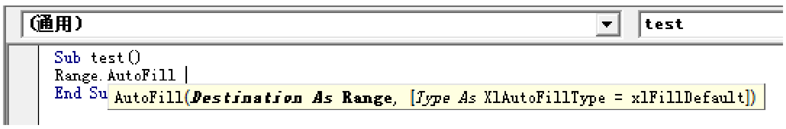


图 9–3　提示参数的数据类型

在图 9-2 中，Worksheet.Protect 方法的每个参数都没有标示数据类型，这是因为所有参数都是变体型。

4. 未正确使用括号

对方法或者过程的参数赋值时，是否使用括号取决于方法或者过程的位置，如果方法或者过程放在句首，那么不能使用括号；当方法或过程作为某语句的参数出现时，则该方法或者过程的参数必须使用括号。

例如将 A1 单元格的值向下填充到 A4，那么 Range.AutoFill 方法的参数不能使用括号，如果按以下形式编写代码，那么执行代码时将会出错：

```
Range("A1").AutoFill (Range("A1:A4"))
```

解决办法是删除 Range("A1:A4")两边的括号。

再如新建一个工作表，将新工作表放置在 Sheet1 之前，那么 Worksheets.Add 的参数不能使用括号，否则无法执行代码。正确的书写方式如下：

```
Worksheets.Add Sheet1
```

不过，如果使用 Msgbox 函数获取新建工作表的名称，那么 Worksheets.Add 将作为 Msgbox 的参数出现，此时 Worksheets.Add 的参数就必须使用括号。代码如下：

```
MsgBox Worksheets.Add(Sheet1).Name
```

掌握此规律后便可以判断何时需要对参数添加括号，何时不需要括号。

9.1.3　变量定义不准确

初学定义变量后，往往会因为不熟悉数据类型的范围而定义失误，从而造成在执行代码时出现“溢出”错误，如图 9-4 所示。

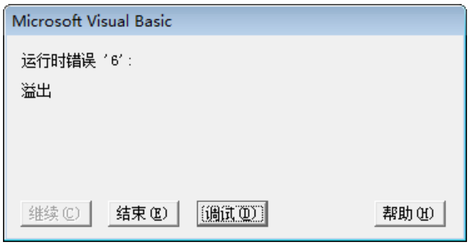


图 9–4　定义变量失误造成“溢出”错误

解决这类错误其实比较简单，而且也不需要花费太多精力去记忆每种数据类型的范围，只需要将本书第 5 章表 5-1 的内容打印出来放在桌上即可，在定义变量前将其当作字典查询。

9.1.4　对象不存在

在执行代码过程中频率最高的错误类型应该是访问的对象不存在，从而导致代码无法执行。对象不存在的范围很广，例如使用 Workbooks.Open 方法打开文件时该路径不存在，或者文件名称书写错误，从而导致读取失败。再如工作簿中只有 2 个表，但以下的代码中却引用了第 3 个表，从而出现“下标越界”错误。

```
Worksheets.Add　Sheets(3)
```

只要出现“下标越界”的错误提示，那么一定表示对象集合或者数组的参数超过了允许的赋值范围。前面所讲的只有 2 个表时却使用代码 Sheets(3)引用第 3 个表，这就是典型的“下标越界”错误。

事实上，找不到文件、路径错误、对象集合的参数超出允许范围、引用单元格时地址书写错误等都具有相同处，那就是访问的对象不存在。

这类错误在实际工作中最常见，需要在编写代码时仔细核对对象名称以及了解被访问对象的允许范围。另外在书写代码时尽量通过计算获得其名称或者许可范围，而不是手动指定名称或序号。

例如在工作簿中有 5 个工作表，最后一个工作表的名称是“Sheet5”，那么需要引用最后一个工作表时不宜采用以下两种方法。

Sheets ("Sheet5")——直接通过名称引用工作表时，当工作表重命名后就会引用失败。

Sheets (5) ——直接指定序号引用最后一个工作表也不具备通用性，增删工作表后都会影响引用结果的正确性。

正确引用最后一个工作表的代码是：

Worksheets (Worksheets.Count)——通过计算得到最后一个工作表的序号，再通过序号引用最后一个工作表。

再如在 D 盘中的“财务报表”中有一个名为“5 月.xlsm”的工作簿，利用代码打开此工作簿时不能直接使用以下代码：

```
Workbooks.Open "D:\财务报表\5 月.xlsm"
```

编程过程中随时都要准备应对意外情况，而不能总是预先假设对象存在。以上代码可以改用以下两种思路处理。思路一的代码如下：

```
If Len(Dir("D:\财务报表\5 月.xlsm")) > 0 Then
 Workbooks.Open "D:\财务报表\5 月.xlsm"
End If
```

此思路是利用 Dir 函数提取文件名称，再用 Len 函数判断文件名的长度是否大于 0，如果大于 0 表示存在该文件。当指定的文件存在时才执行 Workbooks.Open 命令，这就有效地防止了文件不存在时无法执行代码的问题。

第二种思路的代码如下：

```
On Error Resume Next
Workbooks.Open "D:\财务报表\5 月.xlsm"
If Err.Number = 0 Then
 '更多命令
End If
```

代码中使用了错误处理语句“On Error Resume Next”，从而不管文件是否存在，执行 Workbooks.Open 命令都不会弹出错误提示，也不会中断程序。不过当屏蔽错误提示后虽然不弹出错误提示框，但当文件不存在时，VBA 内部仍然会产生错误，仅仅没有通过对话框体现出来而已，所以通过 Err.Number 属性可以捕捉到内部的错误编号，然后可以根据此编号判断 Workbooks.Open 命令是否执行成功。

错误处理语句“On Error Resume Next”和错误对象 Err 等正是本章的重点，本章将具体分析在代码中如何处理错误以及通过 Err 对象执行某些判断。

9.2 错误处理语句

在调用代码过程中总会遇到代码执行出错的问题，有时甚至也会人为地制造错误，从而根据内部的错误编码来执行某些判断。

善用 VBA 的错误处理语句可以预防一些潜在的问题出现，也可以根据不同的错误结果采取不同的补救措施。

9.2.1　详解 Err 对象

VBA 提供了一个 Err 对象，此对象包含代码运行中产生的与错误相关的信息。

Err 对象有 6 个属性和 2 个方法，表 9-1 是 Err 对象的属性说明，表 9-2 是 Err 对象的方法说明。

表 9-1　Err对象的属性

属　性	说　明
HelpContext	包含帮助文件中的主题的上下文 ID
Number	错误编码，Err对象的默认值
HelpFile	表示帮助文件的完整路径
LastDLLError	返回因调用动态链接库 (DLL) 而产生的系统错误号。
Description	包含与对象相关联的描述性字符串
Source	最初生成错误的对象或应用程序的名称

表 9-2　Err对象的方法

方　法	说　明
Clear	清除 Err 对象的所有属性设置
Raise	产生运行时错误

在 Err 对象的 6 个属性中，最常用的是 Number 和 HelpFile，其中前者用于获取错误编码，后者表示此错误编码对应的帮助文件，打开文件可以查看此错误类型的详细解释。

Err 对象的 Clear 方法表示清除错误及清除与错误相关的属性设置。

```
Sub test()                                              '放置位置：模块中
    On Error Resume Next                                '错误时继续执行
    a = 10 / 0                                          '执行 0 做除数的运算
    MsgBox Err.Number & Chr(10) & Err.Description       '提示错误编码及其含义
    Err.Clear                                           '清除错误
    MsgBox Err.Number & Chr(10) & Err.Description       '提示错误编码及其含义
End Sub
```

以上代码首先使用错误处理语句阻止程序弹出错误提示框，因此遇到任何错误都不会中断代码。然后通过表达式“a = 10 / 0”产生一个错误，并用 Err.Number 和 Err.Description 分别提取错误编码和出现错误的原因解释，其结果为错误编码对应“11”，错误原因对应“除数为零”。

后面的 Err.Clear 语句清除了前面产生的错误，所以再次使用 Err.Number 属性获取错误编码时只能返回 0。

Err.Raise 方法则表示人为地制造一个错误，在以下案例中将会用到。

```
Sub 查询错误编码()                                      '放置位置：模块中
Dim ErrNumber As Integer                                '声明变量
'指定待查询的错误编码
    ErrNumber = Application.InputBox("请输入想要查询的错误编码", "错误编码", , , , , , 1)
    On Error Resume Next                                '当代码出错时继续执行
    Err.Raise ErrNumber                                 '人为地产生一个编码等于 ErrNumber 的错误
    '通过 Err.Description 获取错误编码为 ErrNumber 的含义解释。
  MsgBox "您所输入的错误编码的含义是：" & Chr(10) & Err.Description, vbOKOnly, "提示"
End Sub
```

此过程的功能是：创建一个输入框，用户可在其中输入想要查询的错误编码，然后得到该错误编码对应的含义解释。假设执行过程后，在如图 9-5 的对话框中输入 9，然后单击“确定”按钮，将产生如图 9-6 所示的提示框，其中详细地阐述了与错误编码有关的帮助信息。

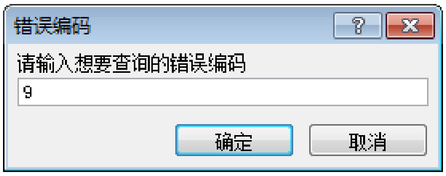


图 9-5　输入待查询的错误编码

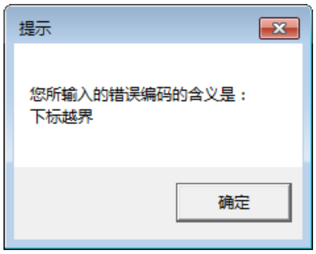


图 9-6　根据编号得到错误原因

以上过程中用到了 VBA 中较多的错误处理语句，接下来将详细展示更多与错误处理相关的语句，并提供案例加深读者的理解。

本例案例文件请参考：..\ 第 9 章\ 9-1 了解 ERR 对象及查询错误编码含义.xlsm

9.2.2　Error 函数详解

Error 函数用于返回对应于已知错误编码的错误信息，其语法如下。

```
Error[(errornumber)]
```

参数 ErrorNumber 表示错误编码，是可选参数。如果参数是有效的错误号，那么函数 Error 返回该错误编码的错误信息，和 Err.Description 的功能一致；如果参数的赋值是一个无效的编码，那么返回“应用程序定义的错误或对象定义的错误”；如果省略参数，那么返回在当前代码中实际产生的错误编码对应的错误信息，若当前代码中没有错误则返回空值。

所以如果需要查询错误编码对应的含义，使用 Error 函数其实比使用 Err 对象的 Description 属性方便得多。

```
Sub 查询错误编码()                    '放置位置：模块中
    Dim ErrNumber As Integer          '声明一个 Integer 型的变量 ErrNumber
    '让用户输入错误编码，并赋值给变量
    ErrNumber = Application.InputBox("请输入想要查询的错误编码", "错误编码", , , , , , 1)
    '报告该编码对应的含义解释
    MsgBox Error(ErrNumber)
End Sub
```

9.2.3　On Error Resume Next 语句

On Error Resume Next 语句是错误处理语句中最有用、使用最频繁的语句。要理解它的含义可以分两个步骤，其一是 On Error 部分，表示当代码执行过程中出现错误时；其二是 Resume Next 部分，它表示转移控制权，将控制权转移到发生错误的语句之后的语句，并在此继续运行，简单而言就是忽略出错的语句，执行下一行代码。

所以 On Error Resume Next 语句在实际工作中通常用于屏蔽代码运行中可能产生的错误提示，避免因某句出错而造成代码中断。

对于初学者而言，代码过程中是否会出错是难以预料的，所以不管是否出错都可以将此错误处理错语句放置在过程首句位置。当熟练操作后，有了足够的经验则可以判断此过程中是否需要

On Error Resume Next 语句，或者可以通过其他办法预防。

以下案例是 On Error Resume Next 语句的典型应用。

要求新建一个工作表，命名为“总表”，如果已经存在“总表”则结束过程，代码如下。

```
Sub 新建总表()                                  '放置位置：模块中
    Dim sht As Worksheet                        '声明一个对象变量，工作表引用工作表
    Set sht = Worksheets.Add                    '新建一个工作表
    On Error Resume Next                        '如果有错误，继续执行下一步
sht.Name = "总表"                               '对新工作表命名
'如果错误编码不等于 0（表示命名不成功，只有已经存在同名工作表时才会失败）
    If Err.Number <> 0 Then
        MsgBox "已经存在总表"                   '提示已经存在“总表”
        Application.DisplayAlerts = False       '关闭提示，避免在删除工作表时弹出提示框
        sht.Delete                              '删除新建的工作表
        Application.DisplayAlerts = True        '恢复提示
    End If
End Sub
```

以上过程的思路是新建一个空表，然后将它命名为“总表”，由于 Excel 不允许工作簿中存在两个“总表”，所以当工作簿中已经有“总表”时，代码“sht.Name = "总表"”会出错，且中断过程。

基于此思路，在“sht.Name = "总表"”之前放置错误处理语句 On Error Resume Next，从而阻止程序弹出错误提示框，避免过程被中断，同时记录错误编码。

接着在条件语句中判断 Err.Number 属性的值是否等于 0，如果不等于 0 则表示已经产生错误，即已存在同名的工作表，那么删除新建的工作表即可。

事实上，实现同等功能还有第二种思路，先借助 On Error Resume Next 语句判断是否存在指定的工作表，然后根据 Err.Number 的值决定后续操作。完整代码如下：

```
Sub 新建总表 2()                                '放置位置：模块中
    Dim sht As Worksheet                        '声明一个对象变量，用于引用工作表
    On Error Resume Next                        '如果有错误，继续执行下一步
Set sht = Worksheets("总表")                    '将变量赋值为“总表”
'如果错误编码等于 0（表示没有错误，已经存在总表的前提下才可能没有错误）
    If Err.Number = 0 Then
        MsgBox "已经存在总表"                   '提示已经存在“总表”
    Else                                        '否则
        Worksheets.Add.Name = "总表"            '新建一个工作表，并命名为“总表”
    End If
End Sub
```

此过程和前一个过程功能一致，但思路大不相同。本例通过判断将名为“总表”的工作表赋值给变量这个过程是否出错，确认工作簿中是否存在“总表”，然后再决定是否新建工作表。

这两个过程的共同点都是先用 On Error Resume Next 语句处理错误，然后计算 Err.Number 的值来判断对象是否存在。

On Error Resume Next 的应用还有很多，在本书前面的章节已经反复出现过，在后面的章节也将有大量的应用。

本例案例文件请参考：..\ 第 9 章\ 9-2 新建总表前的判断.xlsm

9.2.4 On Error GoTo Line 语句

On Error GoTo Line 表示如果代码执行过程中出错，那么跳转到 Line 参数所指定的标签处，执行该标签之后的代码。

On Error GoTo Line 应用于工作时通常采用以下结构：

代码	说明
Sub test()	声明过程名义与参数
On Error GoTo 标签	当错误时执行“标签”后的代码
. . .代码 1. . . . . .代码 2 . .	一组任意代码，可选的
Exit Sub	结束过程，避免执行后面的代码
标签:	设置一个标签
. . . 代码 3. . .	修复错误的代码
Resume [Next]	返回出错的语句继续执行
End Sub	结束过程

假设在以上结构中，“代码 1”在执行过程中出错，那么“On Error GoTo 标签”语句可以使程序跳过“代码 1”，直接执行“标签”后面的“代码 2”。“Resume Next”部分的作用则是返回刚才出错的语句的下一句，并继续执行，即执行“代码 3”。最后在遇到“Exit Sub”时结束过程。

其中“Resume Next”部分中的“Next”是可选的，如果加了“Next”则表示继续执行出错语句的下一句；如果忽略“Next”部分则表示返回出错的语句，重新执行原本出错的那一句代码。

此结构的设计思路对于处理错误有较高的借鉴价值。以下案例便是通过“On Error Goto Line”语句实现处理错误。

要求对如图 9-7 所示的费用表创建一个透视表，实现按季度汇总各项费用，且季度是可调的，即可以通过页字段指定参与汇总的季度。透视表必须保存在名为“透视”的工作表中。

	A	B	C	D	E	F	G	H
1	时间	张签	刘浩华	周克俭	李至民	吴浩天	陈坤	游有之
2	差旅费	1906	1699	2329	1916	1510	2605	2390
3	加汽油	3182	3161	3557	3001	2495	1998	1871
4	过路费	2053	2544	3494	2924	2340	3557	2427
5	办公用品	216	312	288	244	206	319	380

一季度 | 二季度 | 三季度 | 四季度

图 9–7　费用表

根据需求，创建好的透视表需要保存在名为“透视”的工作表中，那么有必要判断是否已经存在“透视”工作表。其次，由于要求季度名称是可调的，那么透视表的类型要采用多重合并计

算数据区域的透视表。

结合 On Error Goto Line 的结构，可以采用以下代码实现。

```
Sub 创建费用透视表()                                    '放置位置：模块中
  Dim Sht As Worksheet                                  '声明一个对象变量，用于引用工作表
  On Error GoTo 不存在                                  '如果有错误，执行标签“不存在”后面的代码
  Set Sht = Worksheets("透视")                          '将名为“透视”的工作表赋值给变量
  Sht.Cells.Clear                     '清除所有单元格的值（两个透视表不能重叠，所以要清除以前的）
  '创建多重合并计算数据区域的数据透视表，其透视表源来 4 个工作表 A1:H5
  '透视表的名称为“费用透视表”，保存在“透视”工作表 A1 开始的区域，版本号为 xlPivotTableVersion14
  ActiveWorkbook.PivotCaches.Create(SourceType:=xlConsolidation, SourceData:= _
     Array(Array("二季度!R1C1:R5C8", "二季度"), Array("三季度!R1C1:R5C8", "三季度"), Array( _
     "四季度!R1C1:R5C8", "四季度"), Array("一季度!R1C1:R5C8", "一季度")), Version:= _
      xlPivotTableVersion14).CreatePivotTable TableDestination:="透视!R1C1", TableName:= _
      "费用透视表", DefaultVersion:= xlPivotTableVersion14
  Exit Sub                                              '结束过程
不存在:                                                 '设置一个标签
  Worksheets.Add(, Sheets(Sheets.Count)).Name = "透视" '新建一个工作表，并命名为“透视”
  Set Sht = Worksheets("透视")                          '将名为“透视”的工作表赋值给变量
  Resume Next                                           '返回前面出现错误的那句代码之后继续执行
End Sub
```

过程中首先将名为“透视”的工作表赋值给变量 Sht，然后使用 ActiveWorkbook.PivotCaches.Create 创建数据透视表缓存，然后使用方法根据缓存创建多重合并计算数据区域的透视表，其中参数 TableDestination 使用“"透视!R1C1"”，表示让透视表保存在“透视”工作表的 A1 单元格开始的区域中，要使用 R1C1 样式，不能用 A1 样式。

假设工作簿中存在“透视”工作表，那么通过“ActiveWorkbook.PivotCaches.Create”方法可以一句代码完成需求。不过实际工作中的工作簿结构并非一成不变，假设工作簿中不存在“透视”工作表，那么创建透视表的操作将无法执行，所以本例使用了 On Error Goto Line 语句，这样在程序出错时可以执行标签“不存在”下方的代码，从而补充一个“透视”工作表，最后再返回出错语句之后执行创建透视表的代码。

换言之，标签“不存在”及其后面的代码就像汽车的备用胎，如果不需要使用，它们就永远闲置，当需要使用时就是救命稻草。本例中出于对代码的完善性的考虑，不能假设工作簿中一定存在“透视表”，而是假设它可能有也可能没有，然后根据有和没有两种情况设置两种处理方式。只有在这种编程思想的指导下才可能让代码更完美，有更好的兼容性。

在本例中，代码的执行结果如图 9-8 所示。

	A	B	C	D	E	F	G	H	I
1	页1	(全部)							
2									
3	**求和项:值**	**列标签**							
4	**行标签**	**陈坤**	**李至民**	**刘浩华**	**吴浩天**	**游有之**	**张签**	**周克俭**	**总计**
5	办公用品	1336	1290	1090	1099	1264	1183	1260	8522
6	差旅费	12712	10732	8541	7887	8490	10112	10785	69259
7	过路费	12514	9325	12283	9892	9295	10173	12712	76194
8	加汽油	10416	10988	12783	11073	11822	12148	13601	82831
9	**总计**	**36978**	**32335**	**34697**	**29951**	**30871**	**33616**	**38358**	**236806**

... 二季度　三季度　四季度　透视

图 9–8　费用透视表

本例案例文件请参考：..\ 第 9 章\ 9-3 创建数据透视表.xlsm

9.2.5 On Error GoTo 0 语句

On Error Goto 0 语句表示禁止当前过程中任何已启动的错误处理机制。

这里所讲的“已启动的错误处理机制”是指 On Error GoTo Line 语句或者 On Error Resume Next 语句，On Error GoTo 0 语句可以抵消这两句代码所产生的作用。

On Error GoTo Line 语句或者 On Error Resume Next 语句都属于错误处理语句，或者说防错语句，它们对防错语句之后的所有代码都生效，任何一句代码产生错误时都会启动设置好的错误处理机制。而有时只希望部分代码有错误时启动已设置的错误处理机制，其他代码则不应用之前所设置的错误处理机制，On Error GoTo 0 语句正是为此而存在。它可以消除以前所做的一切关于错误处理的设置，从而在代码出错时能弹出错误提示框，并通知用户代码有误，以便及时修改。

On Error GoTo 0 语句更多用于调试代码，在调试代码阶段需要在代码出错时弹出错误提示框，从而了解出错的原因。而在实际工作中一般不需要使用，因为它会造成代码出错时弹出提示框，从而导致代码中断。

以前一个案例中的“创建费用透视表”过程为例，代码“On Error GoTo 不存在”主要针对“Set Sht = Worksheets("透视")”而设置的，当此代码有误时会启动“On Error GoTo 不存在”，但是事实上“On Error GoTo 不存在”对过程中的任何错误都会生效，从而造成与“Set Sht = Worksheets("透视")”无关的错误也会执行标签“不存在”后面的代码，这就带来了隐患。所以最好的办法是在“Set Sht = Worksheets("透视")”语句之后添加代码“On Error GoTo 0”，从而既可以避免错误地执行标签“不存在”后面的代码，又可以了解在“Set Sht = Worksheets("透视")”语句之后是否还存在其他的代码错误。

所以在调试代码期间，只要过程中使用了 On Error GoTo Line 语句或者 On Error Resume Next 语句处理错误，就有必要在适当的地方插入 On Error Goto 0，从而测试后面的代码是否存在错误，以便在发现有错误后及时调整代码，消除隐患。

9.2.6 Gosub...Return 语句

GoSub...Return 语句和 On Error GoTo Line 语句的功能相近，都是在符合条件时执行指定标签后面的代码，然后返回继续执行。不同处在于 On Error Goto Line 语句是在发现代码中有任意错误时执行标签后的代码，有可能会执行多次，而 GoSub...Return 语句则是指在符合指定的条件时执行标签后的代码，只判断一次条件。通过以下两个案例可以了解 GoSub...Return 语句和 On Error GoTo Line 语句的差异。

1．标示所有工作表的平均值

要求对工作簿中的所有工作表添加条件格式，将表中的平均值标示出来。由于在工作表可能是空表，或者表中只有文本时会导致计算平均值失败，因此有必要在代码中使用错误处理语句，同时记录所有添加条件格式失败的工作表名称，完整代码如下。

```
Sub 标示所有工作表的平均值()                        '放置位置：模块中
    Dim AverValue, ShtName As String, Rng As Range, Sh As Worksheet    '声明变量
    On Error GoTo line                              '如果有错误，那么执行 Line 后面的代码
    For Each Sh In Sheets                           '遍历所有工作表
        Set Rng = Sh.UsedRange                      '将 Sh 的已用区域赋值给变量 Rng
        AverValue = WorksheetFunction.Average(Rng)  '计算 Sh 表的平均值
        Rng.FormatConditions.Delete                 '清除原有的条件格式
```

```
            '添加条件格式，公式含义是已用区域的第一个单元格的值等于平均值
            Rng.FormatConditions.Add Type:=xlExpression, Formula1:="=" & Rng.Cells(1).Address(0, 0) & "="
& AverValue
            Rng.FormatConditions(1).Interior.Color = 255    '符合条件格式时，标示红色背景
    Next
    '如果 ShtName 的长度大于 0，那么显示在信息框中
        If Len(ShtName) > 0 Then MsgBox "以下工作表设置条件格式不成功" & ShtName
        Exit Sub                                        '结束过程
    Line:                                               '设置一个标签
        ShtName = ShtName & Chr(10) & Sh.Name           '记录执行出错的所有工作表名称
        Resume Next                                     '返回出错的语句之后，继续执行
    End Sub
```

假设工作簿中有如图 9-9 所示的 5 个工作表，其中在“C 班”和“E 班”两个工作表中的成绩数据都是空白的，那么执行以上代码后将会弹出如图 9-10 所示的提示框，表示代码中出现过两次错误，On Error GoTo Line 语句对两次错误都生效。



	A	B	C	D	E	F	G
1	A班成绩表						
2	姓名	语文	数学	化学	政治	英语	体育
3	刘子中	84	50	78	55	55	90
4	诸有光	64	52	65	69	65	98
5	周华章	98	100	64	58	58	82
6	曲华国	70	78	86	66	82	60
7	李文新	74	42	54	72	96	64
8	钟正国	90	70	64	96	82	82
9	陈强生	72	100	78	92	96	92
10	刘文喜	52	76	96	88	84	76
11	梁今明	76	72	68	70	64	52
12	柳三秀	62	98	53	68	68	74

图 9-9　成绩表

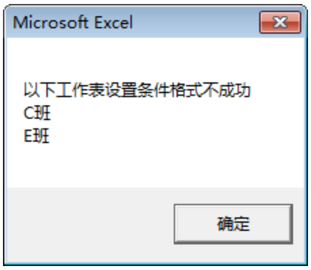


图 9-10　提示执行不成功的表名

一个 GoSub...Return 语句只能执行一次，如果用它代替 On Error GoTo Line 语句实现同样功能，则需要将 GoSub...Return 语句放在循环语句之中，完整代码如下。

```
    Sub 标示所有工作表的平均值 2()                           '放置位置：模块中
        Dim AverValue, ShtName As String, Rng As Range, Sh As Worksheet   '声明变量
        On Error Resume Next                                  '如果有错误继续执行
        For Each Sh In Sheets                                 '遍历所有工作表
            Set Rng = Sh.UsedRange                            '将 Sh 的已用区域赋值给变量 Rng
             AverValue = WorksheetFunction.Average(Rng)       '计算 Sh 表的平均值
             If Err.Number <> 0 Then GoSub Line               '如果有错误，那么跳转到 Line 标签后
             Rng.FormatConditions.Delete                      '清除原有的条件格式
            '添加条件格式，公式含义是已用区域的第一个单元格的值等于平均值
             Rng.FormatConditions.Add Type:=xlExpression, Formula1:="=" & Rng.Cells(1).Address(0, 0) &
"=" & AverValue
             Rng.FormatConditions(1).Interior.Color = 255     '符合条件格式时，标示红色背景
    Next
    '如果 ShtName 的长度大于 0，那么显示在提示框中
     If Len(ShtName) > 0 Then MsgBox "以下工作表设置条件格式不成功" & ShtName
     Exit Sub                                        '结束过程
    Line:                                            '设置一个标签
     ShtName = ShtName & Chr(10) & Sh.Name           '记录执行出错的工作表名称，将它们串连起来
     Err.Clear                                       '清除错误（使 Err.Number 归零）
     Return                                          '返回 GoSub 之后的语句之后，继续执行
    End Sub
```

在使用 GoSub...Return 语句配合 On Error Resume Next 语句时要注意清除错误，即使用 Err.Clear 语句。当第一次产生错误后，Err.Number 的值不再等于 0，执行标签 Line 下方的代码后会返回到前面的 For Each…Next 循环语句，继续执行。如果在标签 Line 之后没有使用 Err.Clear 方法清除错误，那么当进入第二轮循环时，Err.Number 还保留了上一次的错误编码，将会影响代码“If Err.Number <> 0 Then GoSub Line”的判断结果，所以在标签 Line 之后必须通过代码清除错误。

本例案例文件请参考：..\ 第 9 章\ 9-4 GoSub Return 语句和 On Error Goto Line 语句.xlsm

2．拆分工作簿

仍然以前一个案例的文件为例，要求拆分此工作簿，将每个工作表保存为一个单独的 Excel 工作簿，采用每个工作表中 A1 单元格的值作为文件名称，如果 A1 单元格空白造成代码出错，那么将工作表的名称赋值给 A1 再继续执行。

同样通过 On Error Goto Line 语句和 GoSub...Return 语句分别实现，前者代码如下：

```
Sub 拆分工作簿()                                              '放置位置：模块中
    With Application.FileDialog(msoFileDialogFolderPicker)   '创建一个选择文件夹的对话框
        .AllowMultiSelect = False                            '不允许多选
'如果没有单击“取消”按钮（单击“取消”按钮时，Show 的值为 False）
        If .Show = True Then
'将选择的第一个文件夹名字赋予变量 PathName
            PathName = .SelectedItems(1)
'如果路径右边没有"\"，就追加一个"\"
            If Right(PathName, 1) <> "\" Then PathName = PathName & "\"
        End If
    End With
    On Error GoTo Line                                       '如果有错误，那么跳转到标签 Line 之后
    Dim Sht As Worksheet                                     '声明一个对象变量
    For Each sht In Worksheets                               '遍历所有工作表
        Sht.Copy                                             '将工作表复制到新工作簿中
        '将新工作簿保存到前面选择的路径下，以 A1 单元格的字符命名
        ActiveWorkbook.SaveAs Filename:=PathName & Sht.Range("A1").Value & ".xlsx",
FileFormat:=xlOpenXMLWorkbook
        ActiveWindow.Close False                             '关闭工作簿
    Next Sht
Exit Sub                                                     '结束过程，避免执行后面的代码
'设置一个标签，当前面的代码在执行过程中出错时，可跳转至此标签处，执行后面的代码
Line:
    Sht.Range("A1").Value = Sht.Name                         '将工作表名称赋值给 A1 单元格
    '将新工作簿保存到前面选择的路径下，以 A1 单元格的字符命名
     ActiveWorkbook.SaveAs Filename:=PathName & Sht.Range("A1").Value & ".xlsx",
FileFormat:=xlOpenXMLWorkbook
     Resume Next                                             '返回出错的代码的下一行继续执行
End Sub
```

在此过程中，将 On Error GoTo Line 语句放在循环语句之外，但是对循环语句中的任何错误都生效。

Resume Next 表示返回原本出错的代码（另存工作簿）的下一句继续执行，所以当标签 Line 后面的代码修复错误后仍然不会继续执行前面出错的那句代码（另存工作簿），所以本例中有必

要在标签 Line 之后添加另存工作簿的代码。

当然，也可以将 Resume Next 修改为 Resume，那么标签 Line 后面只需要对 A1 赋值即可，不需要执行另存工作簿的操作。

使用 GoSub...Return 语句实现需求的完整代码如下。

```
Sub 拆分工作簿 2()                    '放置位置：模块中
'创建一个选择文件夹的对话框
 With Application.FileDialog(msoFileDialogFolderPicker)
      .AllowMultiSelect = False        '不允许多选
      If .Show = True Then             '如果没有单击“取消”按钮（值为 True 表示单击“确定”按钮）
           PathName = .SelectedItems(1) '将选择的第一个文件夹名字赋予变量 PathName
'如果路径右边没有"\"，就追加一个"\"
           If Right(PathName, 1) <> "\" Then PathName = PathName & "\"
      End If
   End With
   Dim Sht As Worksheet                '声明一个 WorkSheet 型的对象变量
   For Each Sht In Worksheets          '遍历所有工作表
      Sht.Copy                         '将工作表复制到新工作簿中
'如果 A1 单元格是空单元格，那么执行 Line 后面的代码
      If Len(sht.Range("A1").Value) = 0 Then GoSub Line
      '将新工作簿保存到前面选择的路径下，以 A1 单元格的字符命名
      ActiveWorkbook.SaveAs Filename:=PathName & Sht.Range("A1").Value & ".xlsx",
FileFormat:=xlOpenXMLWorkbook
      ActiveWindow.Close False         '关闭工作簿
    Next Sht
Exit Sub
'设置一个标签，当前面的代码在执行过程中出错时，可跳转至此标签处，执行后面的代码
Line:
   Sht.Range("A1").Value = Sht.Name    '将工作表名称赋值给 A1 单元格
   Return                              '返回 GoSub 的下一行继续执行
End Sub
```

GoSub...Return 语句是满足设置的条件时再执行标签 Line 后面的代码，通过 Return 语句返回 GoSub 的下一行（另存工作簿）继续执行，所以在标签 Line 之后不需要执行另存工作簿的命令，只需要修复问题即可，从而让后面的操作得以正常执行。

GoSub...Return 语句和 On Error GoTo line 语句都用于处理意外状况，如果一切正常那么按代码的顺序执行即可，如果存在意外则跳转到标签之后修复问题，然后返回执行其他命令。

如图 9-11 所示，假设本案例中 C 班和 E 班的 A1 单元格是空白的，那么执行过程后将得到如图 9-12 所示结果。



	A	B	C	D	E	F	G
1	B班成绩表						
2	姓名	语文	数学	化学	政治	英语	体育
3	龚月新	80	74	63	82	73	57
4	仇正风	100	90	88	69	55	89
5	郑丽	88	88	80	92	50	60
6	柳洪文	54	55	66	56	50	75
7	张志坚	80	72	92	54	58	84
8	魏郑	88	68	54	85	93	70
9	赵冰冰	54	88	70	76	74	60
10	柳星华	66	54	80	58	96	54
11	谢有金	72	72	94	88	63	84
12	林至文	60	54	50	66	90	65

图 9–11　A1 无数据的成绩表

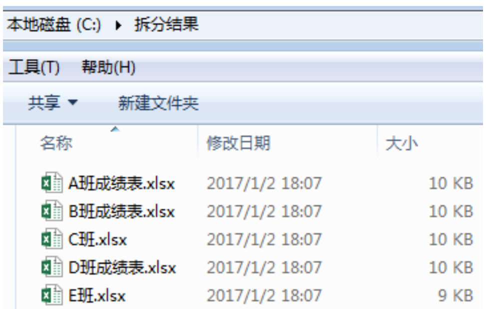


图 9–12　拆分工作簿结果

本例案例文件请参考：..\ 第 9 章\ 9-4 GoSub Return 语句和 On Error Goto Line 语句.xlsm

9.3 案例应用

从理论上讲，规范地设计表格并且正确地使用代码是不会产生代码错误的，然而在实际工作中总是有太多的意外，包括格式不规范、版本不同、路径变化、新用户操作失误等，所以对于开发者而言，每一段代码都需要错误处理措施，避免出现操作不规范或者客观环境变化带来的失误。

当然，有时也会故意制造错误，从而用于获取某些信息。

总之，极有必要在编程中掌握错误处理技巧。

9.3.1 处理错误的常规思路

处理代码错误的常规思路分为三类，一是防范因代码出错而导致程序中断；二是根据错误类型提供相应的解决办法；三是故意利用代码出错来获取某些信息。

1．防范因代码出错而导致程序中断

在默认设置下，VBA 遇到任意错误时都会弹出提示框通知用户，同时中断程序，当解决好问题后重新执行代码，才能正常执行程序。

而事实上一段好的代码应该提前预防问题，并屏蔽错误提示，在 9.2 节中关于 On Error GoTo Line 语句和 Gosub...Return 语句的案例就可以作为很好的参考。

2．根据错误类型提供相应的解决办法

前面说过，在默认设置下，VBA 遇到任意错误时都会弹出提示框通知用户，同时中断程序，然而 VBA 自带的错误提示框并非总能将问题描述清楚，从而导致用户难以正确地判断错在何处。而优秀的代码应该对不同的错误提供不同的解决办法，至少需要提供清晰明了、人性化的错误提示。

假设 A1 和 B1 是空白单元格，执行以下代码必将弹出“溢出”错误。

```
Range("C1") = Range("A1") / Range("B1")
```

修复以上问题的方法是在 A1 和 B1 录入数值，但是无法确保所有用户都能根据“溢出”二字就能明白问题根源在于空白单元格。对于程序员而言，有义务提供更人性化的提示框。

如果 A1 或者 B1 单元格有文本，执行该句代码则将弹出“类型不匹配”错误。很显然，使用者很难根据此提示明白出错的根源。

就本例而言，应修改为如下代码。

```
Sub test()                                      '放置位置：模块中
 On Error Resume Next                           '有错误时继续执行
 Range("c1") = Range("a1") / Range("B1")        '执行除法运算
'假设有错误，那么根据不同的错误编码提供不同的错误解释，从而让用户明白如何处理问题
  If Err.Number = 6 Then
         MsgBox "A1 和 B1 都不能是空单元格", vbCritical, "提示"
  ElseIf Err.Number = 11 Then
        MsgBox "除数不能为 0，请对 B1 单元格赋予非零值", vbCritical, "提示"
  ElseIf Err.Number = 13 Then
        MsgBox "A1 和 B1 单元格都不能是文本，请输入数值", vbCritical, "提示"
  End If
```

```
End Sub
```

当 A1 或者 B1 单元格有文本时，执行以上代码将弹出如图 9-13 所示的提示框；而当 A1 单元格的值是数值并且 B1 单元格空白时则将弹出如图 9-14 所示的提示框。

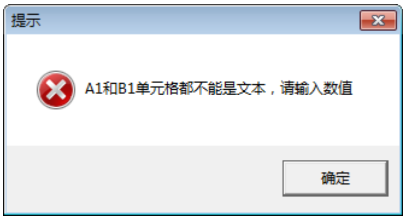


图 9–13　A1 和 B1 单元格有文本时的提示

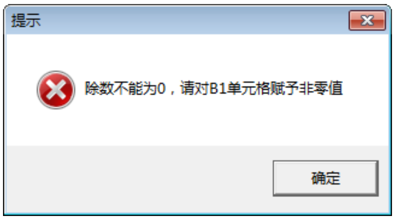


图 9–14　B1 单元格空白时的提示

如果 A1 和 B1 都是空白单元格，那么执行以上代码将弹出如图 9-15 所示的提示框。

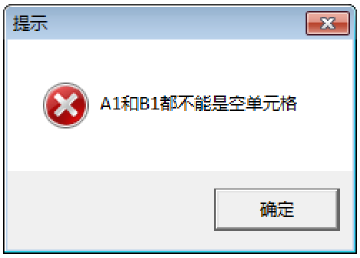


图 9–15　A1 和 B1 都是空单元格时的提示

可见，改进后的错误处理语句代码显得更人性化。

本例案例文件请参考：..\ 第 9 章\ 9-5 提供人性化的错误提示.xlsm

3．故意利用代码出错来获取某些信息

VBA 提供了判断单元格是否空白、单元格是否存在批注、工作表是否被保护、单元格是否有公式、工作表是否处于筛选状态的属性或者函数，但是对于路径是否存在、文件是否打开、是否有指定名称的工作表等却无法判断，面对此类需求只能通过错误编码来判断。

最典型的错误编码应用是将对象赋值给对象变量，如果赋值时出错，那么 Err.Number 属性的值不等于 0，表示对象不存在。在本书前面的章节已有此类思路的大量应用，此处不再举例。

9.3.2　案例应用：按条件定位单元格

【案例要求】Excel 自带查看最大值、最小值、平均值等功能，当选择数值区域时会在状态栏中将它们显示出来。然而 Excel 只能显示值，却不能定位这些值，本例要求开发一个工具来定位工作表中的最大值、最小值、平均值和众数。具体定位哪一种值需要让用户选择，当有多个值符合条件时需要同时选中这些单元格。

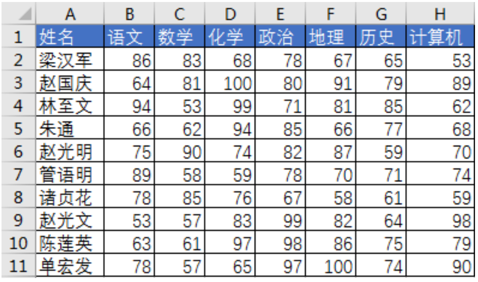
【知识要点】On Error Resume Next 错误处理、Do... Loop Until 循环、Err.Clear 清除错误和 Err.Raise 产生错误。

【程序代码】

```
Sub 按条件定位()                                                    '放置位置：模块中
Dim Rng As Range, ValResult As Byte, TargetVal As Double        '声明变量
Set Rng = ActiveSheet.UsedRange                                  '将本工作表的已用区域赋值给变量 Rng
```

```
    '如果是空表则提示用户，然后结束过程
     If IsEmpty(Rng) Then MsgBox "当前表为空白表格", vbOKOnly, "友情提示": Exit Sub
     '如果工作表中没有数值则提示用户，然后结束过程
    If WorksheetFunction.CountA(Rng) = 0 Then MsgBox "当前表没有数值", vbOKOnly, "友情提示": Exit Sub
    On Error Resume Next   '如果有错误则继续执行下一句
    Do     '启动循环语句
      Err.Clear     '清除错误（避免在录入有误时影响到下一次的判断）
      '弹出对话框让用户指定定位对象，并将用户录入的值赋予变量 ValResult
    ValResult = Application.InputBox("请指定要定位的对象：" & Chr(13) & "1:最大值" & Chr(13) & "2:最小值" &
Chr(13) & "3:平均值" & Chr(13) & "4:众数")
    '如果用户录入的值大于 4，那么仍会生成一个错误，错误编码为 1
     If ValResult > 4 Then Err.Raise 1
    Loop While Err.Number <> 0                        '只要有错误就继续循环下去
    Select Case ValResult                             '以变量 ValResult 的值作为判断条件
    Case 1                                            '当变量 ValResult 的值是 1 时
    TargetVal = WorksheetFunction.Max(Rng)            '将工作表中的最大值赋予变量  TargetVal
    Case 2                                            '当变量 ValResult 的值是 2 时
    TargetVal = WorksheetFunction.Min(Rng)            '将工作表中的最小值赋予变量  TargetVal
    Case 3                                            '当变量 ValResult 的值是 3 时
    TargetVal = WorksheetFunction.Average(Rng)        '将工作表中的平均值赋予变量  TargetVal
    Case 4                                            '当变量 ValResult 的值是 4 时
     TargetVal = WorksheetFunction.Mode(Rng)          '将工作表中的众数赋予变量  TargetVal
    Case Else                                         '否则（变量的值为 0）
    Exit Sub                                          '结束过程
    End Select
    Dim TargetRng As Range, FirstAdd As String        '声明变量
    Set Rng = Cells.Find(TargetVal, , , xlWhole)      '在工作表中查找变量 TargetVal 的值
    If Rng Is Nothing Then                            '如果没找到（找平均值有可能找不到）
       MsgBox "表中不存在 " & TargetVal, vbOKOnly     '提示用户
    Else                                              '否则
     FirstAdd = Rng.Address                           '记录下第一次找到的目标地址
     Set TargetRng = Rng
     Do                                               '循环查找其他单元格
      Set Rng = Cells.FindNext(Rng)                   '查找下一个
    '如果新找到的单元格的地址等于第一次找到的单元格的地址
     If Rng.Address = FirstAdd Then
          TargetRng.Select                            '选择变量 TargetRng 代表的区域
          Exit Sub                                    '然后结果过程
     Else                                             '否则
    '将新找到的目标与上一次的目标合并成一个 Range 对象
         Set TargetRng = Union(TargetRng, Rng)
       End If
    Loop
     End If
    End Sub
```

假设工作表中有如图 9-16 所示的数据，执行以上过程后会弹出如图 9-17 所示的对话框。



	A	B	C	D	E	F	G	H
1	姓名	语文	数学	化学	政治	地理	历史	计算机
2	梁汉军	86	83	68	78	67	65	53
3	赵国庆	64	81	100	80	91	79	89
4	林至文	94	53	99	71	81	85	62
5	朱通	66	62	94	85	66	77	68
6	赵光明	75	90	74	82	87	59	70
7	管语明	89	58	59	78	70	71	74
8	诸贞花	78	85	76	67	58	61	59
9	赵光文	53	57	83	99	82	64	98
10	陈莲英	63	61	97	98	86	75	79
11	单宏发	78	57	65	97	100	74	90

图 9–16　数据

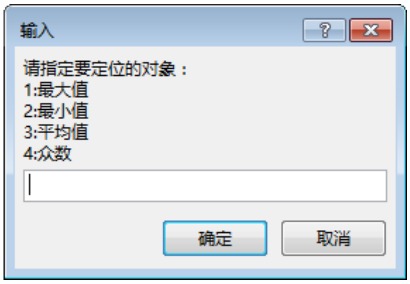


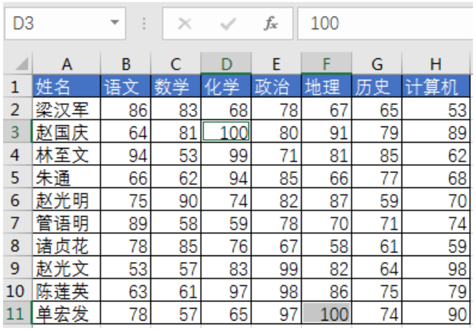
图 9–17　选择定位对象

如果此时输入-1、300 或者任意文本，程序会再次弹出对话框让用户重新输入，因为-1、300 或者任意文本都超出了 Byte 型变量的允许范围。

如果输入 5，程序仍然会再次弹出输入框让用户重新输入，因为在过程中有“If ValResult > 4 Then Err.Raise 1”，它表示只要大于 4 都会产生一个错误值，而由于 Do Loop 循环的结束条件是没有错误，因此输入大于 4 的值时，仍然会弹出对话框。

如果此时输入 1，单击“确定”按钮，那么程序会选中区域中的所有最大值，效果如图 9-18 所示。

如果此时输入 3 并单击“确定”按钮，那么程序会弹出如图 9-19 所示的提示信息，表示工作表中没有某个单元格的值等于平均值。



	A	B	C	D	E	F	G	H
1	姓名	语文	数学	化学	政治	地理	历史	计算机
2	梁汉军	86	83	68	78	67	65	53
3	赵国庆	64	81	100	80	91	79	89
4	林至文	94	53	99	71	81	85	62
5	朱通	66	62	94	85	66	77	68
6	赵光明	75	90	74	82	87	59	70
7	管语明	89	58	59	78	70	71	74
8	诸贞花	78	85	76	67	58	61	59
9	赵光文	53	57	83	99	82	64	98
10	陈莲英	63	61	97	98	86	75	79
11	单宏发	78	57	65	97	100	74	90

图 9–18　定位区域中的最大值

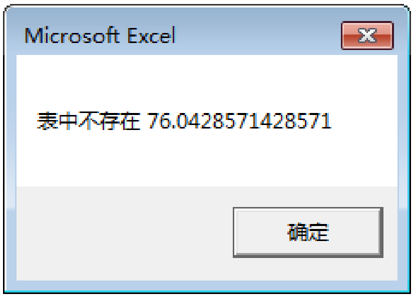
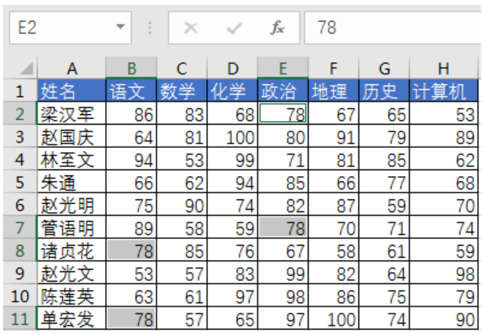


图 9–19　提示未找到平均值

如果输入 4 并单击“确定”按钮，那么程序会瞬间选中工作表中的众数，效果如图 9-20 所示。所谓的众数就是出现次数最多的数值，在该工作表中，数值 78 总共出现了 4 次，它就是本表的众数。



	A	B	C	D	E	F	G	H
1	姓名	语文	数学	化学	政治	地理	历史	计算机
2	梁汉军	86	83	68	78	67	65	53
3	赵国庆	64	81	100	80	91	79	89
4	林至文	94	53	99	71	81	85	62
5	朱通	66	62	94	85	66	77	68
6	赵光明	75	90	74	82	87	59	70
7	管语明	89	58	59	78	70	71	74
8	诸贞花	78	85	76	67	58	61	59
9	赵光文	53	57	83	99	82	64	98
10	陈莲英	63	61	97	98	86	75	79
11	单宏发	78	57	65	97	100	74	90

图 9–20　选中所有众数

【语法补充】

（1）当判断工作表是不是空表时可以用 IsEmpty(ActiveSheet.UsedRange)，返回 True 时表示工作表是空表，否则不是空表；当判断工作表中有没有数值则用 WorksheetFunction.CountA(Rng)，若返回值为 0 则表示没有数值。

（2）当变量被声明为 Byte 型时，它的有效范围是 0~255 之间的整数，如果对此变量赋值为负数、大于 255 的值或者文本，都会出错。因此本例的 Do Loop 循环语句中加了“While Err.Number <> 0”，表示只要有错误存在，就一直循环下去，反复弹出对话框让用户重新输入。

由于用户输入 5 或者 10 时不会出错，它们都在 Byte 型的有效范围之内，因此本例代码中加了“If ValResult > 4 Then Err.Raise 1”，用于在用户录入的值大于 4 时人为生成一个错误值，从而避免终止循环语句。

（3）代码“Err.Clear”的作用是清除错误，它在本例中的作用是清除上一轮循环中留下的错误，避免该错误影响本轮的判断。例如用户第一次在对话框中输入 10，程序会生成一个编号为 1 的错误，如果此时不清除错误，那么第二次循环时输入正确的值 3，由于错误值还在，程序会继续循环下去，让用户重新输入。很显然，这不符合需求，因此必须在用户重新输入之前清除前面的错误值。

（4）当用户在对话框中按下“取消”按钮时，相当于在输入框中输入 0，两者的功能一致，因此并未在 Do Loop 循环中加入 ValResult 的值等于 0 时要如何处理的代码，而是在 Select Case 条件选择中通过 Case Else 来控制处理方式，即用户输入 0 或者按下“取消”按钮时都会结束过程。

（5）最大值、最小值和众数是一定存在的，而平均值可能存在也可能不存在。例如表中有三个数值，分别是 3、5、10，平均值为 6。事实上工作表中没有 6，因此在定位平均值时就会定位失败，为此在编写代码时必须加入“If Rng is Nothing Then”这类判断语句，并提供处理方式，而不要假设一定能定位成功。在编程时必须考虑所有可能存在的情况。

本例案例文件请参考：..\ 第 9 章\ 9-6 按条件定位.xlsm

9.3.3 案例应用：根据选区的文件名批量导入图片

【案例要求】选择存有图片文件名称的区域，然后批量导入图片，将图片保存在文件名称右边的单元格中，如果单元格中的图片名称有误，则将该名称记录下来。

【知识要点】On Error Resume Next Err.Clear Do Loop 和 Application.FileDialog。

【程序代码】

```
Sub 根据选区的图片名称导入对应的图片()            '放置位置：模块中
If TypeName(Selection) <> "Range" Then MsgBox "请选择单元格或单列的区域", vbOKOnly, "友情提示": Exit Sub '如果选择的对象不是单元格，那么提示并结束过程
'声明变量
 Dim path As String, cell As Range, FileName As String, Number As Byte, FileFormat As String
On Error Resume Next                           '有错误时继续执行下一句
Do                                             '启动循环
  err.Clear                                    '清除上一轮循环留下的错误（假设有的话）
  '选择文件格式，支持 PNG、GIF、JPG 和 BMP 格式，所以只允许输入 1、2、3、4
Number = Application.InputBox("输入 1：插入 GIF 图片；" + Chr(10) + "输入 2：插入 PNG 图片；" + Chr(10) + "输入 3：插入 JPG 图片；" + Chr(10) + "输入 4：插入 BMP 图片。", "图片格式", 3, , , , , 1)
'如果输入值不在 1 到 4 之间，那么提示用户，并且人为地制造一个错误
  If Number < 1 Or Number > 4 Then MsgBox "只能输入 1、2、3 或者 4", vbOKOnly: err.Raise 1
```

```
    Loop Until err.Number = 0                              '如果不符合条件就继续循环
    '将输入的序号转换成后缀名
     FileFormat = IIf(Number = 1, ".GIF", IIf(Number = 2, "PNG", IIf(Number = 3, ".JPG", ".BMP")))
     With Application.FileDialog(msoFileDialogFolderPicker)   '创建一个选择文件夹的对话框
    If .Show Then                                          '如果未单击“取消”按钮
    '将用户选择的文件夹地址赋值给变量(如果路径夹右边没有“\”则追加“\”)
      path = .SelectedItems(1) & IIf(Right(.SelectedItems(1), 1) = "\", "", "\")
     Else                                                  '否则
      Exit Sub                                             '结束过程
    End If
    End With
    Application.ScreenUpdating = False                     '关闭屏幕更新，从而加快代码执行速度
    For Each cell In Selection                             '遍历选区
    If Len(cell.Value) > 0 Then                            '如果单元格不是空单元格
       '向 cell 单元格中录入图片，图片的图路为 path & cell.Text & FileFormat，由文件夹、文件名称和后缀名三
部分组成
       '插入图片时自动将图片放在右方单元格中，图片的左边距、上边距、宽度和高度都由 cell.Offset(0, 1)决定
       '插入图片后，通过 With 语法引用此图片对象，因为后面还需要调用此对象
          With ActiveSheet.Shapes.AddPicture(path & cell.Text & FileFormat, False, True, cell.Offset(0, 1).Left,
cell.Offset(0, 1).Top, cell.Offset(0, 1).Width, cell.Offset(0, 1).Height)
            If err.Number = 1004 Then                      '如果错误编码为 1004(未找到文件此就会出错)
               FileName = FileName & cell & Chr(10)        '那么记录文件名称
               err.Clear                                   '清除错误，避免影响下一次判断
            Else                                           '否则
              .ShapeRange.LockAspectRatio = msoFalse       '取消图片的“锁定纵横比”
              .Placement = xlMoveAndSize                   '让图片的位置与大小随单元格变化而变化
            End If
          End With
      End If
    Next cell
    '如果这里 FileName 已经赋值过，那么将它显示在提示框中
    If Len(FileName) > 0 Then MsgBox "以下文件未找到" & Chr(10) & FileName, 64, "友情提示"
     Application.ScreenUpdating = True                     '恢复屏幕更新
    End Sub
```

假设工作表中有如图 9-21 所示的数据，选择 A2:A10 区域后执行以上过程，首先会弹出如图 9-22 所示的输入框。假设文件夹中的图片格式是 JPG 格式，那么输入 3 并单击“确定”按钮，程序会弹出一个“浏览”对话框，接下来选择保存图片文件的文件夹名称，然后单击“确定”按钮，程序会自动将文件夹中与选中区域同名的图片导入到工作表中。如果某些单元格中的图片名称有误，无法导入同名的图片文件，那么在程序结束前会弹出一个提示框，提示所有未导入成功的文件名称，如图 9-23 所示。

导入的图片显示在图片名称的右边一列，其高度、宽度、左边距和上边距完全与单元格对应，效果如图 9-24 所示。不过由于所有导入的图片已经设置为可以自由调整大小，且修改单元格时图片相应的移动，所以可以随意修改单元格的宽度或高度来改变图片的显示效果。图 9-25 是修改单元格高度与宽度后的效果。



图 9–21　图片名称

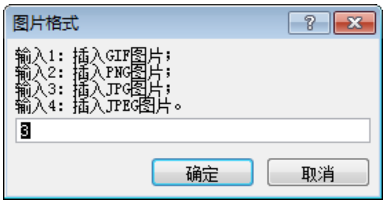


图 9–22　选择图片格式

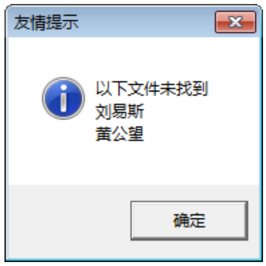


图 9–23　提示未导入成功的名称



图 9–24　导入结果



图 9–25　调整结果

【语法补充】

（1）本例中 On Error Resume Next、err.Clear 和 Do…Loop Until 配合应用的思路和前一个案例大同小异，但这种方法提供了一个出错说明，帮助用户了解程序出错的问题根源。

（2）将用户录入的值转换成图片后缀名时，采用了与前一个案例不同的思路，前例采用 Select Case，本例采用 IIf，两者的写法不同、功能一致。对于编程新手而言，Select Case 的用法应该更好理解。

（3） Shapes.AddPicture 方法表示插入图片，其参数为图片路径。如果参数有误或者原来的图片被删除了，那么无法操作成功，会产生编码为 1004 的错误。所以本例中根据错误编码判断是否插入成功，不成功则记录当前的图片名称。

在记录图片名称后必须马上使用 err.Clear 方法清除错误，否则错误编码遗留下来后会影响下一轮的判断。

本例案例文件请参考：..\ 第 9 章\ 9-7 根据选区的文件名批量导入图片.xlsm

9.3.4　案例应用：一键屏蔽错误值

【案例要求】一键屏蔽工作表中的错误值。

【知识要点】On Error Resume Next、Err.Clear 和 Err.Number。

【程序代码】

```
Sub 一键屏蔽错误值()                                                        '放置位置：模块中
  Dim Rng1 As Range, Rng2 As Range, Rng3 As Range, cell As Range   '声明变量
  Dim Bool As Boolean, Bool2 As Boolean
On Error Resume Next                                                         '代码出错时继续执行下一步
```

```
    '将常量错误赋值给变量 Rng1
     Set Rng1 = ActiveSheet.UsedRange.SpecialCells(xlCellTypeConstants, 16)
     If Err.Number = 0 Then Bool = True              '如果没有错误将变量 Bool 赋值为 True
    Err.Clear                                        '清除借误
    '将公式错误赋值给变量 Rng2
        Set Rng2 = ActiveSheet.UsedRange.SpecialCells(xlCellTypeFormulas, 16)
        If Err.Number = 0 Then Bool2 = True          '如果没有错误将变量 Bool 赋值为 True
        Err.Clear                                    '清除错误
        If Bool And Bool2 Then                       '如果 Bool 和 Bool2 都是 true
            Set Rng3 = Union(Rng1, Rng2)             '将 Rng1 和 Rng2 合并再赋值给 Rng3
        ElseIf Bool And Bool2 = False Then           '如果 Bool 值为 True,但 Bool2 值为 False
            Set Rng3 = Rng1                          '将 Rng1 赋值给 Rng3
        ElseIf Bool = False And Bool2 Then           '如果 Bool 值为 False,但 Bool2 值为 True
            Set Rng3 = Rng2                          '将 Rng2 赋值给 Rng3
        Else                                         '否则
            End                                      '结束过程
        End If
        For Each cell In Rng3                        '遍历 Rng3
             cell.Font.Color = cell.Interior.Color '将单元格的字体与背景调为相同颜色
        Next cell
End Sub
```

这种屏蔽错误值的思路为：借用循环语句逐一将单元格的字体颜色修改为背景色，所以执行以上过程后，将所有错误值所在的单元格屏蔽起来。

【语法补充】

（1）通过 Range.SpecialCells 方法定位对象时，如果对象不存在则弹出错误提示，同时中断程序，而不是返回 Nothing，因此必须在过程中使用 On Error Resume Next 语句防错，并通过 Err.Number 属性判断是否存在被定位的对象。

（2）由于工作表中可能存在多种背景色，所以本例采用循环语句逐一修改单元格的字体颜色。如果确定工作表中所有单元格都未设置背景色，那么可以使用以下 3 句代码即可。

```
    Sub 一键屏蔽错误值()      '放置位置：模块中
    On Error Resume Next
     ActiveSheet.UsedRange.SpecialCells(xlCellTypeConstants, 16).Font. ColorIndex = 2
    ActiveSheet.UsedRange.SpecialCells(xlCellTypeFormulas, 16).Font.ColorIndex = 2
    End Sub
```

（3）如果只是需要在打印时屏蔽错误值，而不是永远都屏蔽起来，那么使用以下代码更简便。

```
    ActiveSheet.PageSetup.PrintErrors = xlPrintErrorsBlank
```

本例案例文件请参考：..\ 第 9 章\ 9-8 一键屏蔽错误值.xlsm

9.4　课后思考

1．用什么办法可以确保在代码出错时不弹出错误提示框？假设过程中超过 4 句代码，用什么办法能让最后 3 句代码在出错时才弹出错误提示框？

2．用代码将活动工作簿保存在 D 盘“文件备份”文件夹中，文件名称为“备份.xlsm”。在代码中需要判断 D 盘是否存在“文件备份”文件夹和“备份.xlsm”文件，以便提供足够的错误处理机制。

3．用实例证明 Resume Next 与 Resume 的区别。

4．假设工作簿中有 10 个工作表，请用 A1:A10 单元格的值分别对 10 个工作表命名，要让代码具有通用性，包括在 A1:A10 区域空白时、A1:A10 存在重复值或者非法字符时，提供必要的提示。

5．Err.Clear 语句适宜在什么情况下使用?

第 10 章　使用数组提升程序效率

数组的存在价值就是让代码提速。

数组和非数组的差异只在于数据的保存与读取方式不同，虽然操作这些数据的方法或者函数并没有不同，但是保存与读取上的差异却使 VBA 代码在处理数据时实现了质的飞跃。在完成相同工作时，使用数组比非数组的效率有可能提升几倍乃至几十倍，数组对于 VBA 而言举足轻重。

读者可以在掌握本章知识后利用数组的知识去优化前面几个章节中的部分案例，也会发现很多程序都有提速的空间。可以说掌握数组后，便完全进入了一片新天地。

本章要点

◆ 基本概念

◆ 数组函数

◆ 案例分析

10.1　基本概念

对于程序效率而言，数组有着神奇的功效，不过关于数组的基础理论也不少，明白这些理论才可能在使用数组时得心应手。

10.1.1　何为数组

数组就是连续可索引的具有相同内在数据类型的元素的集合，数组中的每一元素具有唯一索引号。简单而言，数组就是一组相同类型的数据集合。

可以通过索引号访问数组中的每个元素，也可以随时修改数组中的任意元素。

数组支持一维到六十维，不过常用的是一维数组和二维数组。

一维数组、二维数组和区域有太多的相似点，所以通常可以借助区域来理解数组，因为数组存在于内存中，显得虚无缥缈，而区域则比较形象化。

事实上，数组和区域之间是相互依存的关系，工作中会经常将区域中的数据导出到数组中，当在数组中处理完毕后，又需要将数据从数组导出区域中。本章及后面的章节都会有大量的关于区域与数组相互转换的应用案例。

10.1.2　数组的特点

数组有以下三个特点。

1．包含多个元素

顾名思义，数组包含一组数据，单个数据无法构成数组，也没有必要使用数组。

事实上，由于数组的功能是提速，而数据越多越能体现数组的优势，所以工作中使用数组时往往涉及大量的待运算的数据。

2．读取速度快

计算机在读取数据时，从软盘中读取的速度最慢，其次是光盘、U 盘、硬盘，最快的是内存。而读取 Excel 工作表的单元格中的数据时相当于硬盘级速度，读取数组中的数据则相当于内存级速度，所以数组的运算速度快于区域的运算速度。

在使用数组时，通常将区域中的值读取到内存中，然后再针对数组执行各种运算，运算完毕后再根据需求将结果写入相应的区域或者单元格中。

换言之，使用数组就是尽量减少读取单元格的次数，替之以读取内存，从而提高代码执行效率。

3．不能常驻内存

内存中的数据生命周期不长，不像工作表那样可以长期保留数据。

VBA 中数组的载体其实是指数组变量或者集合（Collection），它们都会在结束过程或者关闭工作簿后自动消失，所以数组仅用于临时保存数据，在内存中处理数据后需要再将数据导出到区域中。数据无法长期驻留于数组中。

10.1.3 一维数组

VBA 中的数组包含一维、二维、三维……直到六十维，常用的是一维数组和二维数组。

在理解一维数组之前，先了解一下一维区域。

1．一维区域

一维区域是指单列多行或者单行多列的区域。图 10-1 是一维纵向区域，包含 5 个单元格；图 10-2 是一维横向区域，包含 3 个单元格。

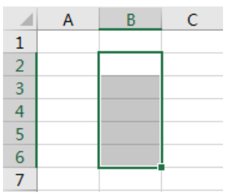


图 10–1 一维纵向区域

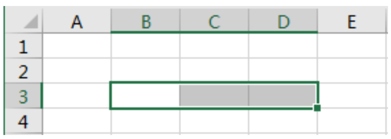


图 10–2 一维横向区域

可以利用转置函数 Transpose 将横向区域的值转换到纵向区域中，也可以将纵向区域的值转换到横向区域中。图 10-3 即为 Transpose 函数转置区域的应用结果。

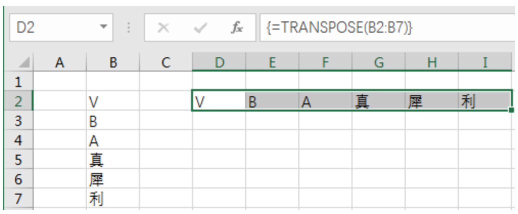


图 10–3 将纵向一维区域的值转换到横向一维区域中

在图 10-4 中包含一个可以产生一维横向数组的公式，输入数组公式的方式是选择 B2:F2 区域，然后录入以下公式，最后按下【Ctrl+Shift+Enter】组合键即可。

```
={1,3,5,7,10}
```

图 10-5 中包含一个可以产生一维纵向数组的公式，公式如下。

```
={"语文";"数学";"地址";"化学"}
```

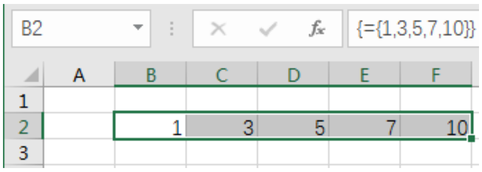


图 10–4　一维横向数组公式

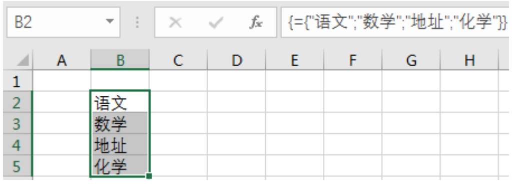


图 10–5　一维纵向数组公式

一维数组公式的特点是公式同时产生多个结果，必须同时选择多行一列或者多列一行的区域然后输入公式，并且按【Ctrl+Shift+Enter】组合键结束。其中在横向数组的元素与元素之间采用逗号作为分隔符；而在纵向数组的元素与元素之间采用分号作为分隔符。

2．一维数组

一维数组和前面的一维区域具有类似的特性，不过只有一维横向数组，没有一维纵向数组。VBA 将具有多行单列的数组定义为二维数组，通过 Transpose 函数将它转换成横向数组后就成了一维数组。

公式中通过花括号产生的一维常量数组可以应用于 VBA，在 VBA 中利用方括号或者 Evaluate 函数可以将它转换成 VBA 的数组。

如果改用 VBA 实现与图 10-4 和图 10-5 的同等效果可以分别采用以下代码：

```
Range("B2:F2").Value = [{1,3,5,7,10}]
Range("B2:B5").Value = [{"语文";"数学";"地址";"化学"}]
```

第一句代码使用的是一维横向数组，第二句代码使用的是二维纵向数组，只不过这个二维数组只有一列。

利用数组在区域中批量输入数据时有较大的优势，不需要逐个向单元格写入值，而是同时在区域中的每个单元格产生数据。

3．产生数组的方式

产生数组的方式较多，主要体现为以下 5 种。

（1）手动指定数组元素

前面所讲的“[{"语文";"数学";"地址";"化学"}]”就是手动指定数组的每个元素，从而生成常量数组。

也可以用 Evaluate 函数代替方括号生成数组，不过这和方括号的写法稍有不同。

当采用 Evaluate 函数代替方括号时，由于其参数是文本，所以需要添加引号，代码如下。

```
Range("B2:F2").Value = Evaluate("{1,3,5,7,10}")
Range("B2:B5") .Value = Evaluate("{""语文"";""数学"";""地址"";""化学""}")
```

使用 Evaluate 函数时要注意两点，其一是花括号前后需要使用半角的双引号；其二是当花括号里面的数组元素是文本时，它原本自带的单个双引号要改写成两个双引号。例如数组中的“"语文"”改写成“""语文""”。

手动指定数组元素还可以使用 Array 函数，示例如下。

```
Range("B2:F2") = Array(1, 3, 5, 7, 10)
```

要注意的是 Array 函数只能创建一维横向数组，不能使用分号创建纵向数组，也不能生成二维数组。

（2）引用区域

在区域中包含多个数据，将区域的值读取到内存中就形成了数组。以下代码先声明一个变体型的变量，然后将区域的值赋予变量，此时变量 Arr 就变成了数组。

```
Dim Arr As Variant
Arr = Range("A1:A10").Value
```

引用区域所产生的数组都是二维数组，不管区域是横向的还是纵向的，也不管包含多少行、多少列，这和区域中的一维与二维概念稍有分别。

（3）使用转置函数 Transpose

工作表函数 Transpose 可将区域转换成数组，不过 Transpose 不是 VBA 函数，调用 Transpose 时需要加上其父对象名称 WorksheetFunction。

以下过程可将一维纵向的区域 A1:A10 转换成一维横向的数组。

```
Dim Arr As Variant
Arr = WorksheetFunction.Transpose（Range("A1:A10"))
```

更多关于 Transpose 函数的详细说明请参阅本章 10.2.4 节。

（4）使用字典 dictionary

字典对象 dictionary 提供了一个 Add 方法，它可以将多个数据逐个写入到字典对象的关键字或者条目中，从而形成数组，同时也提供 Items 方法和 Keys 方法导出数组。

（5）使用 Split 函数

Split 函数可以将字符串按指定的分隔符转换成下标为 0 的一维数组，示例如下。

Split("四川,重庆,朝天门", ",")——生成包括四川、重庆和朝天门三个元素的一维数组。

Split("12 元 88 元 0.55 元", "元")——生成包括 12、88、0.55 和空字符串四个元素的一维数组

在本章后面在将会详细讲述 Split 函数的语法和应用案例。

在同一个数组中，每个元素的生成方式必须一致，不能部分元素是引用，部分元素是常量。例如以下代码会产生错误，因为数组中 4 个元素的生成方式不统一。

```
Range("A1:D1") .Value = [{1,2,range("A1"),"VBA"}]
```

10.1.4　二维数组

二维数组是指多行多列的数组，其行数和列数大于 1，没有上限。不过数组是为工作而服务的，而工作中的数据受工作表的区域大小限制，所以在实际工作中使用数组时，数组的行数与列数上限都小于工作表的最大行数与列数。

A1:B5 是一个二维区域，以下公式可以产生二维数组。

```
={"序号","目录";1,"Sheet1";2,"Sheet2";3,"Sheet3";4,"Sheet4"}
```

选择 A1:B5 后输入以下公式，然后按【Ctrl+Shift+Enter】组合键，运行效果如图 10-6 所示。

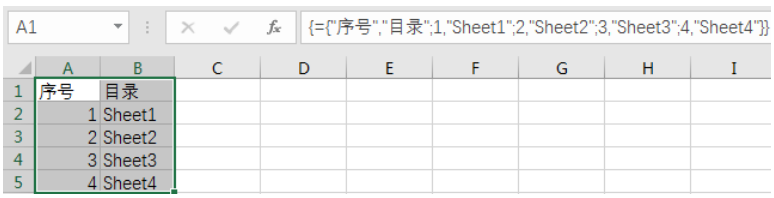


图 10–6　能产生二维数组的公式

如果用 VBA 中的数组输入与图 10-6 中相同的数据，可使用以下代码。

```
Range("A1:B5").Value = [{"序号","目录";1,"Sheet1";2,"Sheet2";3,"Sheet3";4,"Sheet4"}]
```

对比二维数组和一维数组可以发现，二者的差异在于行数、列数不同。

二维数组有以下特点。

其一，在同一行中元素与元素之间的分隔符是逗号，换行时则使用分号。

其二，每一行的元素个数必须一致，同一列中的元素个数也必须一致。如果某行或者某列缺少一个元素，那么整个数组中所有元素都会失效。例如在前面的代码中删除“Sheet4”后，代码将产生如图 10-7 所示的错误值。

解决办法是对该元素赋值为空文本占位，代码如下。

```
Range("A1:B5") = [{"序号","目录";1,"Sheet1";2,"Sheet2";3,"Sheet3";4,""}]
```

代码运行效果如图 10-8 所示。

	A	B
1	#VALUE!	#VALUE!
2	#VALUE!	#VALUE!
3	#VALUE!	#VALUE!
4	#VALUE!	#VALUE!
5	#VALUE!	#VALUE!

图 10–7　缺少元素时产生错误值

	A	B
1	序号	目录
2	1	Sheet1
3	2	Sheet2
4	3	Sheet3
5	4	

图 10–8　使用空文本占位

其三，可以借助 Transpose 函数将二维数组的行与列转置。例如以下代码可以让 2 列 5 行的数组转换成 2 行 5 列。

```
Range("A1:E2") .Value= WorksheetFunction.Transpose(  _
[{"序号","目录";1,"Sheet1";2,"Sheet2";3,"Sheet3";4,"Sheet4"}])
```

代码运行效果如图 10-9 所示。

	A	B	C	D	E	F
1	序号	1	2	3	4	
2	目录	Sheet1	Sheet2	Sheet3	Sheet4	

图 10–9　转置二维数组

10.1.5　数组的参数

可以把一维数组和二维数组看作区域引用，这有助于理解数组和数组的参数。

Range ("B2:B5")是一个一维区域，使用索引号可以获取区域中单个单元格的值，例如以下代码可以获取 Range ("B2:B5")区域中第二个值。

```
MsgBox Range("B2:B5")(2)
```

也可以采用双索引号引用一维区域中的第二个值，代码如下。

```
MsgBox Range("B2:B5")(2,1)
```

二维区域和一维区域一样，既可以使用单索引号也可以使用双索引号引用其中单个单元格的值。

Range ("A1:C5")是一个二维区域，以下代码可以获取该区域中第 3 行第 2 列的值。

```
MsgBox Range("A1:C5")(3, 2)
```

以下代码可引用 Range ("A1:C5")中第 6 个单元格的值。

```
MsgBox Range("A1:C5")(6)
```

如果能理解以上从 Range 对象中获取单个单元格的值的思路，那么理解数组的参数就容易多了。

在提取数组中的单个元素时也有单索引号和双索引号之分。不过相较于区域引用的索引号，数组中的索引号数量有着更严格的规定。

在 VBA 中，引用一维数组中的单个元素时只能使用单索引号，示例如下。

```
Sub test()                              '放置位置：模块中
    Dim Arr1                            '声明一个变体型变量
    Arr1 = Evaluate("{1,2,5,8}")        '将 Evaluate 函数产生的一维数组赋值给变量
    MsgBox Arr1(1)                      '获取数组中第 1 个值
    Dim Arr2                            '声明一个变体型变量
    Arr2 = [{1,2,5,8}]                  '将方括号产生的数组赋予变量
    MsgBox Arr2(1)                      '获取数组中第 1 个值
    Dim Arr3                            '声明一个变体型变量
    Arr3 = Array(1, 2, 5, 8)            '将方括号产生的数组赋予变量
   '获取数组中第 1 个值，不过 Array 产生的数组下标为 0，所以第 1 个值是 2，第 0 个值为 1
    MsgBox Arr3(1)
End Sub
```

在引用二维数组中的单个元素时必须使用双索引号，示例如下。

```
Sub test2()                             '放置位置：模块中
     Dim Arr                            '声明变量，必须采用变体型，否则无法将区域的值赋予此变量
     Arr = Range("A1:C5").Value         '将区域的值写入变体型变量，此后变量成为真正的数组
     MsgBox Arr(3, 2)                   '获取数组中第 3 行第 2 列的值
End Sub
```

以下代码用于从 Evaluate 函数创建的二维数组中取值，也必须使用双索引号。

```
Sub test3()                             '放置位置：模块中
  MsgBox Evaluate("{""序号"",""目录"";1,""Sheet1"";2,""Sheet2"";3,""Sheet3"";4,""Sheet4""}")(2, 2)
End Sub
```

10.1.6 声明数组变量

数组变量的声明方式和其他任何变量的声明方式都有区别，其中涉及较多的规则。

数组总是包含多个元素，在赋值时有一次性赋值所有元素和分次逐个赋值这两种方法，而这两种方法决定了数组声明方式的不同。

如果是一次性对数组赋值，那么声明变量只能使用变体型变量，而不能使用数组变量，在将一个数组赋值给变体型变量后，变量自然会转变成数组变量。

```
Sub test()                              '放置位置：模块中
     Dim Arr1, Arr2                     '声明两个变体型变量
     Arr1 = Evaluate("{1,2;3,4;5,6}")   '将函数产生的数组赋值给变量，从而转换成数组
Arr2 = Range("A1:B10").Value            '将区域的值赋予变量，从而转换成数组
End Sub
```

以上过程中声明了两个变体型变量，当将数组赋值给两个变量后，变量都成了数组。

第一个变量的赋值方式是利用 Evaluate 函数产生一个二维数组，然后赋值给变量，此时变量将成为二维数组。

第二个变量的赋值方式是引用区域的值，然后生成数组。代码“Arr = Range("A1:B10").Value”也可以简写为“Arr = Range("A1:B10")”，二者含义一样，因为 Range 对象的默认属性是 Value，省略属性名称时仍然调用它的 Value 属性值。

不过并非任何情况下省略属性名称时都会调用 Range 对象的 Value 属性值，通过 Set 语句对变量赋值时例外。

以下过程中变量 Arr1 和 Arr2 都是变体型变量，但由于赋值方式不同，对变量的数据类型的影响不同，赋值时调用的对象也不同。代码“Arr1 =Range("A1:B10")”在赋值时调用的是区域的值，即 Range 对象的 Value 属性；代码“Set Arr2 = Range("A1:B10")”在赋值时调用的是区域本

身，即 Range 对象，所以赋值后变量 Arr2 由变体型变量转换成 Range 型。

```
Sub test()                                  '放置位置：模块中
    Dim Arr1, Arr2                          '声明两个变体型变量
    Arr1 = Range("A1:B10")                  '将区域的值赋予变量，从而转换成数组
  '通过 Set 语句将区域赋值给变量，此时变体型变量将成为 Range 型变量
   Set Arr2 = Range("A1:B10")
   '在信息框中显示两个变量的数据类型，从而判断赋值方式对变体型变量的影响
    MsgBox "Arr:" & TypeName(Arr1) & Chr(13) & "Arr2:" & TypeName(Arr2)
End Sub
```

所以，需要产生数组时，切记在赋值时不要使用 Set 语句。

如果需要对数组变量逐个赋值，那么在声明变量时应直接声明为数组变量，即在变量名称后面添加括号，并在括号中指定数组的维数和每一维的上标与下标。

数组变量允许使用变体型以外的数据类型。

例如声明一个有 5 个元素的一维数组，那么应该采用以下两种方式。

```
Dim Arr1(1 To 5)
Dim Arr2(4)
```

第一种声明方式指定了数组变量 Arr1 是一个一维数组，其下标是 1、上标是 5。

第二种声明方式指定了数组变量 Arr2 也是一个一维数组，其下标采用默认值 0，上标是 4，同样包含 5 个元素。

也可以在声明数组变量时指定其数据类型，代码如下。

```
Dim Arr1(1 To 5) as String
Dim Arr2(4) as Long
```

“Arr1(1 To 5)”和“Arr2(4)”两种方式声明的数组变量有何区别呢?本质上是没有区别的，仅在调用数组中的元素时采用的索引号不同。

在默认设置下，变量 Arr1(1 To 5)所代表的数组的下标为 1，而变量 Arr2(4)所代表的数组的下标为 0。

调用下标为 1 的数组中第 1 个值时，其索引号用 1；而调用下标为 0 的数组中第 1 个值时，其索引号用 0。

以下过程包含对下标为 0 和 1 的两个数组的声明、赋值和取值过程，可以通过此过程了解下标不同的数组在赋值及取值上的差异。

```
Sub test()                                  '放置位置：模块中
  '定义两个包含 5 个元素的一维数组变量，以及一个 Byte 型变量
   Dim Arr1(4), Arr2(1 To 5), i As Byte
   For i = 1 To 5                           '从 1 到 5  循环执行 5 次
  '对数组的 i–1 个元素赋值为 i（由于 i 的最小值是 1，而数组的下标是 0，所以减 1）
         Arr1(i - 1) = i
         Arr2(i) = i                        '对数组的第 i 个元素赋值为 i
   Next i
   MsgBox Arr1(0)                           '获取 Arr1 数组的第 1 个值，即下标为 0 的值
'获取 Arr2 数组的第 1 个值，即下标为 1 的值，和 Arr1 的第一个值完全相等
   MsgBox Arr2(1)
End Sub
```

如果声明二维数组变量，那么分别在括号中指定每一维的上标与下标即可。以下是声明二维数组变量的几种方式。

```
Dim Arr1(4, 2)
```

以上代码声明了一个二维数组变量：第一维的下标为 0、上标为 4；第二维的下标为 0、上

标为 2，在数组中有 15 个元素。

```
Dim Arr2(1 To 5, 1 To 3)
```

以上代码声明了一个二维数组变量：第一维的下标为 1、上标为 5；第二维的下标为 1、上标为 3，在数组中有 15 个元素。

```
Dim Arr1(1 To 10, 6 To 10)
```

以上代码声明了一个二维数组变量：第一维的下标为 1、上标为 10；第二维的下标为 6、上标为 10，在数组中有 50 个元素。不过工作中极少有人会用这种方式声明变量。

现提供一个数组的应用案例，读者可以从中看到数组的价值，以及在不同需求下采取的数组声明方式与赋值方式。

【案例要求】在图 10-10 中包含 3000 个学生的成绩，要求根据 B 列的成绩判断是否“及格”，如果不及格则在成绩的右方单元格中返回“不及格”三字。

	A	B	C
1	姓名	成绩	
2	赵	50	
3	钱	88	
4	孙	72	
5	李	50	
6	周	89	
7	吴	75	
8	郑	74	
9	王	54	
10	冯	61	
11	陈	47	

图 10-10　成绩表

【知识要点】不同需求下的数组变量的声明方式。

【程序代码】

为了展示应用数组的优势，特采用两种方式实现案例需求，应用数组的代码如下。

```
Sub 对小于 60 分成绩进行注释()                '采用数组法，执行时间不到下一个过程的 50%
    Dim i As Integer, Tim As Single           '声明两个变量
  '声明两个数组变量（第一个数组变量需要一次性赋值，第二个数组变量则逐一赋值）
    Dim Arr1, Arr2(1 To 3000, 1 To 1)
    Tim = Timer                               '获取当前时间
    '将成绩赋予数组变量，后续读取时不再读取单元格，而是从内存中取值
    Arr1 = Range("B2:B3001")
      For i = 1 To 3000                       '循环数组
          '如果数组中某元素小于 60，则对第二个数组对应的值赋值为“不及格”
            If Arr1(i, 1) < 60 Then Arr2(i, 1) = "不及格"
      Next i
      Range("C2:C3001") = Arr2                '将第二个数组的值导出到 C 列
      MsgBox Format(Timer - Tim, "0.00") & 秒  '报告执行时间
End Sub
```

【代码分析】

1. 以上过程中使用了 Arr1 和 Arr2 两个数组变量，第一个变量是直接调用区域的值，数组的维数和每一维的上标、下标由区域决定，所以声明变量时不需要括号，以及指定上标与下标。

第二个数组变量需要根据成绩决定赋值方式，不可能一次性赋值完成，所以声明变量时需要指定数组的维数和上标、下标。

2. 本例中数组的应用主要体现在两个方面，其一是将成绩区域转换成数组，然后在判断成绩是否小于 60 时改为读取数组中的值，而不是读取单元格中的值，因为读取数组的速度远快于读取单元格；其二是将原本需要逐个写入到单元中的数据写入到数组变量 Arr2 中，最后再一次

性将数组的值导出到单元格。换言之，以前需要多次向单元格写入值的操作改为只写入一次，从而提升操作速度。

3．本例中数据变量 Arr1 用于提升读取数据的速度，将原本读取单元格 3000 次的操作修改为只读取一次，以后的读取操作都针对数组 Arr，从而提升效率；数据变量 Arr2 的作用则是提升写入的速度，将原本需要写入 900 多次的操作转换成只向单元格中写入一次。

为了方便读者比较，接下来提供不使用数组的过程代码：

```
Sub 对小于 60 分成绩进行注释 2()              '直接读写单元格（放置位置：模块中）
    Dim i As Integer, Tim    Single          '声明变量
    Tim = Timer                              '获取当前时间
    For i = 2 To 30001                       '从 2 开始至 30001 结束
        '如果小于 60，则在右边单元格标注“不及格”
        If Cells(i, 2) < 60 Then Cells(i, 3) = "不及格"
    Next i
    MsgBox Format(Timer - Tim, "0.00 秒")    '报告时间
End Sub
```

读者可以反复执行以上两个过程，比较它们的执行时间差异。

本例案例文件请参考：..\ 第 10 章\ 10-1 数组与非数组的效率比较.xlsm

10.1.7　动态数组与静态数组的区别

10.1.6 节中提到在声明其中一种数组变量时需要指定数组的维数和每一维的元素个数，其实那是理想状态下的用法，实际工作中往往在声明变量时不确定每一维的元素个数，而是在声明变量之后使用代码计算得到的，因此在声明变量时就无法指定其每一维的元数个数，这就引申出另一个知识点——动态数组与静态数组的区别。

所谓的静态数组是指维数一定、每一维的元素个数一定的数组，书面解释是“在代码执行期间不能修改数组的上界（最后一个元素的索引号）的数组”。前一个案例中的数组 Arr2(1 To 3000, 1 To 1)在声明和使用时都固定了数组的维数和元素个数，所以它是一个静态数组。

动态数组在工作中应用较频繁，例如前一个案例的要求修改为罗列出所有不及格人员的姓名和成绩，那么由于声明变量时无法获知不及格的人数，所以需要采用动态数组。

1．声明动态数组变量的方法

声明动态数组比较特别，需要分多个步骤完成。先用 Dim 语句声明一个带空圆括号（没有维数和下标）的数组变量，然后用 ReDim 语句重置数组的维数和上标、下标。

通常 ReDim 语句会配合变量使用，而变量的值通过计算得来，这也正是动态数组的存在价值，即根据实际情况决定数组中最末维的上标，而不是预先固定上标。

以下代码是最典型的动态数组声明方法：

```
Dim Arr(), i As Integer
i = Cells(Rows.Count, 1).End(xlUp).Row
ReDim Arr(1 To i, 1 To 1)
```

第一句代码声明了一个没有维数和下标的数组变量；第二句代码对变量 i 赋值为 A 列最后一个非空行的行号；第三句代码利用 ReDim 语句重新指定数组变量的维数和每一维的下标、上标。

2．重置数组时不保留原值

前面讲到 Redim 语句在重置数组变量时有一个特点——在数组变量已经赋值时会清除数组

中的原有数据。通过以下代码可以印证 Redim 的这个特性。

```
Sub test()                          '请对 A1:B5 区域写入数据后再测试代码（放置位置：模块中）
    Dim Arr()                       '声明数组变量
    Arr = Range("A1:B5").Value      '对数组变量赋值
    MsgBox Arr(2, 2)                '获取变量中第 2 行、第 2 列的值
    ReDim Arr(1 To 3, 1 To 4)       '重新指定数组的维数和下标
    MsgBox Arr(2, 2)                '再次获取数组中的第 2 行第 2 列的值
End Sub
```

在代码中第一次获取数组中的值时可以得到对应于 B2 单元格的值，当使用 ReDim 语句重置变量后将清除数组中的一切数据，所以第二个 Msgbox 函数只能取得空值。

通过以下案例可以更深入地了解动态数组的应用。

【案例要求】图 10-10 中包含 3000 个学生的成绩，要求罗列出所有成绩不及格的学生姓名、成绩，结果存放在以 D1 开始的区域。

【知识要点】ReDim。

【程序代码】

由于符合条件的人数需要通过计算才能得出，；因此本例宜用动态数组，代码如下：

```
Sub 罗列所有不及格人员姓名与成绩()                              '放置位置：模块中
     Dim i As Integer, j As Integer, Rng As Range, Arr1, Arr2()  '声明变量
     Set Rng = Range([A2], Cells(Rows.Count, 2).End(xlUp))      '将成绩区域赋值给 Range 对象变量
   '重新指定数组变量的维数（包含二维，第一维的上标由工作表函数 CountIf 计算而来）
    ReDim Arr2(1 To WorksheetFunction.CountIf(Rng, "<60"), 1 To 2)
   '将 Rng 对象变量的值赋予数组变量（静态变量）
Arr1 = Rng.Value
'循环，起始值为 1，终止值为数组第一维的下标
For i = 1 To UBound(Arr1)
          '如果数组 Arr1 的第二维第 i 行中的值小于 60
           If Arr1(i, 2) < 60 Then
                  '那么累计变量 j，j 代表符合条件的数据个数
                    j = j + 1
                  '将数组 Arr1 的第一维第 i 行的值输出到数组 Arr2 中（即姓名）
                   Arr2(j, 1) = Arr1(i, 1)
                  '将数组 Arr1 的第二维第 i 行的值输出到数组 Arr2 中（即成绩）
                  Arr2(j, 2) = Arr1(i, 2)
             End If
    Next i
    '一次性将数组 Arr2 的值输出到 D、E 列，从第一行开始，结束行由变量 j 决定
     Range("D1:E" & j) = Arr2
End Sub
```

【代码分析】

（1）在以上过程中声明了两个数组变量，其中 Arr1 用于保存成绩区域的值，一次性赋值即可，因此只需要“Dim Arr1”语句。

数组变量 Arr2 用于保存不及格的成绩信息，在声明变量时无法获知不及格人数，所以先用 Dim 声明一个不指定维数的上标、下标的数组，然后另起一行利用 ReDim 语句重新指定数组的维数、上标及下标。

也就是说，在 Dim 语句声明数组变量时，上标和下标只能是数值，而使用 ReDim 语句重置数组的上标和下标时，既可以使用数值也可以使用变量，还可以使用包含函数的表达式等。

（2）在本例中 For Next 循环的初始值是 1，终止值使用了 UBound(Arr1)，它表示数组 Arr1 的第一维的元素个数，可以理解为二维数组的行数，也就是本例中的成绩个数。在本书 10.2 节中将会详细介绍 Ubound 函数的语法和应用。

（3）在本过程中，变量 j 的作用是计算符合条件的人数，即在循环语句中每找到一个不及格的成绩就对变量 j 累加一次，变量 j 的值代表当前需要处理的不及格的成绩的序号，它刚好对应数组 Arr2 中需要写入的值的顺序，所以赋值时分别采用“Arr2(j, 1) = Arr1(i, 1)”和“Arr2(j, 2) = Arr1(i, 2)”。

（4）将数组的值一次性输出工作表时，直接用 Arr2 即可，不需要索引号。在本例中，代码“Range("D1:E" & j) = Arr2”表示将 Arr2 的值导出以 D1 开始，以 E 列变量 j 行结束的区域中。其中变量 j 的值等于数组 Arr2 的行数。在图 10-11 中，D 列和 E 列的值是程序的执行结果。

	A	B	C	D	E
1	姓名	成绩		赵	50
2	赵	50		李	50
3	钱	88		王	54
4	孙	72		陈	47
5	李	50		褚	53
6	周	89		卫	53
7	吴	75		沈	51
8	郑	74		韩	51
9	王	54		许	42
10	冯	61		吕	43
11	陈	47		张	56

图 10–11　罗列不及格学生信息

本例案例文件请参考：..\ 第 10 章\ 10-2 罗列不及格学生姓名与成绩.xlsm

注意：ReDim 语句可以反复地改变数组的元素以及维数，但是不能使用 ReDim 语句修改数组的数据类型，除非该数组变量声明为变体型，且没有使用括号。

3．重置数组时保留原值

前一个案例中在循环语句之前使用了工作表函数 CountIf 计算符合条件的成绩数量，所以可以通过 ReDim 语句直接重置数组的上标。事实上在有些情况下是无法用函数或者其他任何办法计算出最终符合条件的数据个数的，只能在循环过程中逐一判断，反复计算符合条件的数据个数，并同步更新数组的上标，以确保数组足以存放所有符合条件的值。这就引申出 VBA 数组中的另一个概念——在重置数组变量时保留原值。

VBA 中重置数组变量的上标且能保留原值的语句是 ReDim Preserve。

◆　在声明数组变量时使用了空括号

如果在声明数组变量时使用了空括号，例如“Dim Arr()”，那么 ReDim Preserve 语句有以下特点。

（1）可以修改一次数组变量的维数。

（2）可以修改一次任意维数的下标。

（3）可以反复修改最末一维的上标。

◆　在声明数组变量时指定了维数

在声明数组变量时指定了维数，例如“Dim Arr(1 to 2,1 to 5)”或者“Dim Arr2(5,6)”，那么 ReDim Preserve 语句有以下特点。

（1）不可以修改数组变量的维数。

（2）不可以修改任意维数的下标。

（3）可以反复修改最末一维的上标。

以上特点表明 ReDim Preserve 在使用中限制较多。

不过，虽然 ReDim Preserve 语句并不完美，但是工作中还是需要它的，它能给工作带来太多的帮助。

接下来看看 ReDim Preserve 的工作模式。

```
Sub test()                                  '放置位置：模块中
    Dim Arr()                               '声明数组变量
  '重新指定数组的维数和上标（ReDim Preserve 能多次修改最后一维的上标
  '但只能改变一次数组的维数）
   ReDim Preserve Arr(1 To 2, 1 To 2)
   Arr(1, 1) = 1                            '数组赋值，方便后面的测试
   Arr(1, 2) = 2
   Arr(2, 1) = 3
   Arr(2, 2) = 4
   ReDim Preserve Arr(1 To 2, 1 To 3) '重新指定数组最末一维的上标（第一维的上标不可以修改）
  '获取数组中的第 2 行第 2 列的值（可以发现在重置最末一维的上标后，原数据还在）
   MsgBox Arr(2, 2)
End Sub
```

以上过程中先利用 Dim 语句声明一个数组变量，然后利用 ReDim Preserve 语句重置数组的维数和上标、下标。ReDim Preserve 语句第一次重置数组变量时可以修改其维数和第一维的下标、上标，而后只能修改最末一维的上标。

在过程中第一次重置数组变量后，数组包含两维，每一维有两个元素，分别对数组的四个元素赋值。接着再次使用 ReDim Preserve 语句重置数组，这一次只能修改最末一维的上标，如果将代码“ReDim Preserve Arr(1 To 2, 1 To 3)”修改为“ReDim Preserve Arr(1 To 3, 1 To 2)”，那么执行代码时会产生“下标越界”错误。

在过程的最后，使用 Msgbox 函数获取二维数组中第 2 行第 2 列的值，结果是 2，表示 ReDim Preserve 语句在重置数组时会保留重置前的数据。

ReDim Preserve 语句的这个特性极其有用，下面的案例就展示了它的独特优势。

【案例要求】图 10-12 的工作簿中包含 3 个工作表，每个工作表的学生人数不确定，要求罗列出所有班级中不及格的学生信息，包括姓名和成绩，结果显示在“不及格”工作表中。



	A	B	C
1	姓名	成绩	
2	朱通	81	
3	周锦	67	
4	曹值军	57	
5	吴国庆	74	
6	蒋有国	61	
7	严西山	100	
8	陈文民	56	
9	陈冲	51	
10	郑君	67	

图 10–12　三个班级的成绩表

【知识要点】ReDim Preserve 和 Transpose。

【程序代码】

```
Sub 罗列多个班级的不及格学生姓名与成绩()        '放置位置：模块中
```

```
        Dim sht As Worksheet, Item As Integer, TargetCount As Integer, Arr, Arr2() '声明变量
        For Each sht In Worksheets                                      '遍历所有工作表
            Arr = sht.UsedRange.Value                                   '将 sht 的已用区域的值赋予数组变量
           '开始循环，初始值为 2（第一行是标题），终止值由数组上标决定
            For Item = 2 To UBound(Arr)
                If Arr(Item, 2) < 60 Then                               '如果数组的第 2 列的值小于 60
                    TargetCount = TargetCount + 1                       '记录符合条件的数据个数
                   '重置数组变量 Arr2（由于只有最末一维的上标才允许多次修改，所以先将数组
                   '变量重置为横向的包括两行多列的数组，当对赋值完成后再转置为纵向即可）
                  ReDim Preserve Arr2(1 To 2, 1 To TargetCount)
                  '将数组 Arr1 中的第一列的值（姓名）赋值给 Arr2 的第 1 行第 TargetCount 列
                  '（其实是最后一列，因为数组的列数总是等于符合条件的成绩个数）
                  Arr2(1, TargetCount) = Arr(Item, 1)
                 '将数组 Arr1 中的第 2 列的值（成绩）赋值给 Arr2 的第 2 行第 TargetCount 列
                  Arr2(2, TargetCount) = Arr(Item, 2)
              End If
          Next Item
       Next
       On Error Resume Next                                             '遇到错误时继续执行下一句
      '如果 TargetCount 大于 0（即存在不及格的成绩）
       If TargetCount > 0 Then
           '屏蔽提示，避免删除工作表时弹出提示框
            Application.DisplayAlerts = False
           '删除“不及格”工作表（假设有的话，此处用于预防，避免下一句代码出错）
            Worksheets("不及格").Delete
            Application.DisplayAlerts = True                            '恢复提示
           '添加一个新工作表，保存在所有工作表之后，且命名为“不及格”
            Worksheets.Add(after:=Worksheets(Worksheets.Count)).Name = "不及格"
            Worksheets("不及格").Range("a1:b1") = [{"姓名","成绩"}]   '写入标题
            Worksheets("不及格").Range("A2").Resize(TargetCount, 2) = WorksheetFunction.Transpose(Arr2)
'一次性将数组的值导出到工作表中
           '对已用区域添加边框
            Worksheets("不及格").UsedRange.Borders.LineStyle = xlContinuous
        End If
    End Sub
```

代码运行效果如图 10-13 所示。



	A	B	C	D
1	姓名	成绩		
2	计尚云	55		
3	罗至贵	56		
4	徐大鹏	50		
5	梁兴	54		
6	陈随机	59		
7	曲华国	57		
8	刘文喜	54		
9	魏郑	54		

图 10–13　不及格学生信息

【代码分析】

1．在本例中使用了两个数组，一个用于提升读取数据的速度，一个用于提升写入数据的速度。其中数组 Arr 用于存放工作表中的学生信息，当区域中的值导入到数组中后，读取数组替代

了读取单元格，从而大幅缩短读取时间。而数组 Arr2 则用于保存所有符合条件的数据，借用此数组可以一次性将数据写入到新工作表“不及格”中，从而缩短写入数据的时间。

2．由于在执行循环语句之前无法获得所有班级中不及格的人数，所以在声明数组变量时无法指定其每一维的上标，因此可以首先声明一个不指定维数的数组变量，然后在循环语句中使用 ReDim Preserve 语句为数组变量重新分配维数和上标、下标。

在第一次使用 ReDim Preserve 时，允许修改数组变量的维数和上标、下标，但是在第二次使用 ReDim Preserve 时则只能修改数组最末维的上标。所以本例采用“ReDim Preserve Arr2(1 To 2, 1 To TargetCount)”而不是“ReDim Preserve Arr2(1 To TargetCount, 1 To 2)”。

3．按照需求，动态数组变量 Arr2 应该包含两列（第一列存放姓名，第二列存放成绩），行数则由不及格人数决定，即数组的第一维是动态的，第二维的上标固定为 2。但是鉴于 VBA 的规定，不允许反复修改第一维上标，所以本例采用了变通的方式，声明数组变量行数为 2、列数由不及格成绩的数量决定，向数组中写入数据后将变成横向数组。

为了避免最后在将数组的值导出工作表时方向与要求不一致，所以本例采用了工作表函数 Transpose 将其转置方向，然后导出工作表。

4．本例中的“On Error Resume Next”和“Worksheets("不及格").Delete”语句都是为了处理错误而存在，虽然图 10-12 的案例文件中并没有工作表“不及格”，但是编程时仍然有必要通过代码处理一些意外情况，先假设可能存在同名的表，然后通过代码来处理该表。在此思想指导下产生的代码才可能完善，读者在编程过程中有必要养成这种习惯。

5．过程中最后一个条件语句的作用同样是处理错误，避免在表中没有不及格成绩时也创建一个名为“不及格”的工作表，而且执行一些不必要的操作。在编程时要考虑所有的可能情况，不能一切按正常或者最常出现的情况处理，否则代码将留下太多漏洞，也浪费执行时间。

本例案例文件请参考：..\ 第 10 章\ 10-3 罗列多个班级的不及格学生姓名与成绩.xlsm

10.1.8 释放动态数组的存储空间

数组变量中保存了大量的数据，当确定不再使用时须要释放变量的值。数组变量和普通变量的生命周期一致。过程级变量的生命周期最短，在过程结束时就自动释放该变量的值，所以不需要手动释放变量的值。模块级变量和工程级变量属于公有变量，需要关闭 Excel 或者手动释放变量，变量占用的空间才会释放出来。

而手动释放数组变量的值有两种方法，其一是 Erase 语句，其二是 End 语句。

1．Erase 语句

Erase 语句的语法如下。

```
Erase arraylist
```

参数 arraylist 代表数组，一次只能释放一个数组变量的值。假设模块中有三个公有数组变量 Arr1、Arr2 和 Arr3，那么可采用以下三句代码释放所有数组变量的值。

```
Erase Arr1
Erase Arr2
Erase Arr3
```

2．End 语句

当需要释放的数组变量太多时，End 语句可以一次性释放所有变量的值，所以 End 语句的

便利性和高效性超过 Erase 语句。

不过使用 End 语句有可能产生失误，因为它是释放所有变量的值，包括一切公有变量和私有变量，而且不仅仅是针对数组变量。所以在使用 End 语句前需要仔细斟酌。

10.2　数组函数

VBA 提供了诸多内置的数组函数，可以方便地操作数组。在应用数组前有必要了解这些函数的功能和语法。

10.2.1　用函数创建数组

VBA 提供了专用于创建数组的函数 Array 和 Evaluate，它们都能创建数组。

1．Array 函数创建一维横向常量数组

Array 函数只能创建一维常量数组，其语法如下：

```
Array(arglist)
```

参数 arglist 是一个用逗号隔开的数据表，表中的值分别用于给数组的各个元素赋值。例如 Array(1,2,3)表示创建一个包含三个元素的一维横向常量数组，第一个元素的值为 1，第二个元素的值为 2，第三个元素的值为 3。

假设需要向工作表中 A1、B1、C1、D1 单元格输入标题“姓名”、“工号”、“产量”和“不良率”，那么不需要分四次赋值，而是直接利用 Array 函数产生一个包含四个元素的一维数组，然后将数组赋值给区域，代码如下。

```
Range("A1:D1").Value = Array("姓名", "工号", "产量", "不良率")
```

使用数组对区域进行赋值比逐个对单元格赋值的效率更高，不过 Array 函数的缺点是只能创建一维横向常量数组，如果需要创建纵向的二维数组，那么可以利用转置函数 Transpose 将 Array 创建的数组转换方向，例如在 A1:A4 区域产生“一月”、“二月”、“三月”和“四月”，那么可以采用以下代码。

```
Range("A1:A4") = WorksheetFunction.Transpose(Array("一月", "二月", "三月", "四月"))
```

2．Evaluate 函数创建横向与纵向的常量数组

Evaluate 函数比 Array 函数强大得多，它既可以创建一维常量数组，也可以创建二维常量数组，还可以通过引用区域的值创建内存数组。不过“Range("A1:A10").Value”这种模式也可以将区域的值转换成数组，所以在工作中更多的时候是用 Evaluate 函数创建常量数组。

在使用 Evaluate 函数创建一维常量数组时，其参数是文本字符串，而文本字符串中需要使用花括号，在花括号中包含创建数组的数据列表。如果创建只有单行的一维横向常量数组，那么元素与元素之间采用逗号作为分隔符，如果创建单列的纵向常量数组，那么元素与元素之间采用分号作为分隔符。示例如下。

Evaluate("{1,3,5,7,9}")——创建包括五个元素的一维横向常量数组。

Evaluate("{2;4;6;8}")——创建包括四个元素的二维纵向常量数组（只有一列但仍是二维数组）。

图 10-14 与图 10-15 比较形象地展现了以上两个数组的区别。

	A	B	C	D	E
1	1	3	5	7	9
2					

图 10-14　一维横向常量数组

	A	B
1	2	
2	4	
3	6	
4	8	
5		

图 10-15　二维纵向常量数组

如果数组中的元素是文本，那么需要在文本的两边添加双引号。但是由于花括号外面已经有双引号了，所以数组元素的每个双引号都要改为两个双引号，这样 VBA 才能正常识别，示例如下。

Evaluate("{2;""VBA"";6;8}")——表示创建一维纵向数组，其中第二个元素是字符串“VBA”，它的前后各采用了 2 个半角的双引号，如果采用单个双引号则会产生编译错误。

Evaluate 函数创建的一维横向常量数组和 Array 函数创建的稍有不同——Evaluate 函数创建的一维横向常量数组下标为 1，Array 函数创建的下标为 0。以下过程创建了两个值相同的一维横向常量数组，不过在访问其中的数据时需要采用不同的索引号，因为它们的下标不同。

```
Sub test()                          '放置位置：模块中
    Dim Arr, Arr2
    Arr = Array(1, 2, 3, 4, 5, 6)
    Arr2 = Evaluate("{1,2,3,4,5,6}")
    MsgBox Arr(2)                   '取索引号为 2 的值，结果为 3（下标为 0）
    MsgBox Arr2(2)                  '取索引号为 2 的值，结果为 2（下标为 1）
End Sub
```

以上代码采用两个函数创建了具有相同元素的数组，不过由于下标不同，所以同样是访问索引号为 2 的元素，得到的答案并不相同。

Array 函数创建的数组下标总是 0，因此索引号为 0 的元素才是该数组中的第一个值。

3．Evaluate 函数创建多行多列的二维常量数组

Evaluate 函数创建二维常量数组和一维常量数组的方式相似，只是在需要换行的地方添加分号即可，例如将一维常量数组 Evaluate ("{1,2,3,4,5,6}")修改为 2 行 3 列的二维常量数组，代码修改为“Evaluate ("{1,2,3;4,5,6}")”即可，在本书 10.1.4 小节中也有关于二维常量数组的介绍。

需要特别强调的是，代码“Evaluate("=Sheet2!A1:A5")”创建的其实是 Range 对象，而不是数组，“Evaluate("=Sheet2!A1:A5").value”才能创建数组。

10.2.2　获取数组元素

VBA 允许调用工作表函数 Index 从数组中获取单个值，语法如下。

```
WorksheetFunction.Index(arraylist, Row_num,Column_num)
```

其中参数 arraylist 表示数组，参数 Row_num 表示行数，参数 Column_num 表示列数。此处的行数、列数和数组的索引号稍有不同。

行数与列数的最小值是 1，而数组的索引号（下标）可能是 0 或 1，也可能是 100 或者 555 等，这取决于声明数组变量的方式。

```
Sub test()    '区分不同数组的下标，以及了解 Index 函数从数组中取值的特性（放置位置：模块中）
    Dim Arr, Arr2, Arr3(5 To 10)        '声明三个数组变量
    Arr = Array(1, 2, 3, 4, 5, 6)       '使用 Array 函数对第一个数组赋值（其下标为 0）
    Arr2 = Evaluate("{1,2,3,4,5,6}")    '使用 Evaluate 函数对第二个数组赋值（其下标为 1）
    Arr3(5) = 1                         '对第三个数组的元素逐个赋值（其下标为 5）
```

```
        Arr3(6) = 2
        Arr3(7) = 3
        Arr3(8) = 4
        Arr3(9) = 5
        Arr3(10) = 6
        MsgBox WorksheetFunction.Index(Arr, 1, 3)     '获取第一行第三列的值
        MsgBox WorksheetFunction.Index(Arr2, 1, 3)    '获取第一行第三列的值
        MsgBox WorksheetFunction.Index(Arr3, 1, 3)    '获取第一行第三列的值
End Sub
```

在以上过程中采用三种方式创建了三个数组，它们的下标分别是 0、1、5，不过使用 Index 函数读取数组中的元素时全按行、列数计算，不用计较上标与下标的值。

事实上，Index 函数也可以从数组中提取整行或者整列数据。

以下代码先将一个二维数组赋值给变量 Arr1，然后利用 Index 函数将此数组的第二行提取出来赋值给变量 Arr2，最后将变量 Arr2 的值输出工作表。

```
Sub 批量提取数组中整行的值()
  Dim Arr1, Arr2                               '声明两个变量
  Arr1 = [{1,2,3;4,5,6;7,8,9}]                 '对 Arr1 赋值为二维数组，包含三行三列
  Arr2 = WorksheetFunction.Index(Arr1, 2)      '将 Arr1 的第 2 行提取出来赋值给变量 Arr2
  Range("a3:c3") = Arr2                        '将 Arr2 的值输出到 A1:C1 中，此时 Arr2 是一个一维的数组
End Sub
```

执行以上过程后，会在 A1:C1 区域中产生 4、5、6 这三个数字。

10.2.3　判断变量是否为数组

VBA 提供了判断变量是否为数组的函数——IsArray。

当函数 IsArray 的返值是 True 时，表示参数是数组，否则不是数组。

在实际工作中，IsArray 函数的价值极小，几乎不用。

10.2.4　转置数组

工作表函数 Transpose 在数组应用中有一个很有用的函数，不过它属于工作表函数，所以调用此函数需要指定父对象 WorksheetFunction，其全称为“WorksheetFunction.Transpose”。

转置函数 Transpose 主要有三方面功能：转换数组方向、将区域引用转换成数组、将任意的只有单行或者单列的二维数组转换成一维数组。

1．转换数组方向

在 VBA 的数组应用中经常需要转置数组的方向，特别是动态数组的应用。

```
Sub test()
        Dim Arr As Variant
        Arr = Evaluate("{""V"", ""B"", ""A""; ""It"", ""Is"", ""Good""}")
        Range("A1:C2") = Arr
        Range("E1:F3") = WorksheetFunction.Transpose(Arr)
End Sub
```

在以上过程中利用 Evaluate 函数创建了一个二维常量数组，它的每个元素的排序方式为如图 10-16 中 A1:C2 区域所示的效果；当使用 Transpose 函数转置数组后则得到如图中 E1:F3 区域所示的效果。

	A	B	C	D	E	F
1	V	B	A		V	It
2	It	Is	Good		B	Is
3					A	Good

图 10-16　转置数组前后效果比较

2. 将区域引用转换成数组

使用区域引用作为 Transpose 函数的参数时将会生成数组，不过数组的方向与区域本身的行列方向相反。

例如：Range("A1:E1")是横向的区域引用，其类型为“Range”，而 WorksheetFunction.Transpose(Range("A1:E1"))的类型则是“Variant()”，此时它是一个纵向的数组。

3. 将单行或者单列的二维数组转换成一维数组

Transpose 在处理只有单行或者单列的数组时比较特别，它在改变数组方向的同时还可以改变数组的维数。

当一维的区域引用赋值给变体型变量后，此变量所代表的数组是一个二维数组，其第一维和第二维的下标都是 1，需要使用双索引号才能引用数组中的元素。使用 Transpose 函数能将它们转换成一维数组。

在以下过程中使用了两个变体型变量，分别将一维横向区域和一维纵向区域赋值给两个变量，从而使两个变量变成二维数组，然后再通过 Transpose 函数将它们转换成一维数组。

```
Sub 将二维区域转置成横向的一维数组()            '放置位置：模块中
    Dim Arr1, Arr2                             '声明变体型变量
    Arr1 = Range("A1:A5").Value                '此时 Arr1 是 5 行 1 列的二维数组
    Arr1 = WorksheetFunction.Transpose(Arr1)   '此时 Arr1 是一维数组

    Arr2 = Range("A1:E1").Value                '此时 Arr2 是 1 行 5 列的二维数组
 '此时 Arr2 是一维数组
    Arr2 = WorksheetFunction.Transpose(WorksheetFunction.Transpose(Arr2))
End Sub
```

此外，Transpose 函数还有一个比较神奇的功能：Join 和 Filter 函数都不支持直接引用区域的值而产生的数组，但是利用 Transpose 函数将区域转置后产生的数组将允许作为 Join 和 Filter 函数的参数参与运算。

例如将 A1:A10 区域的值合并到 B1 单元格中，以下代码无法执行。

```
Sub 合并区域的值()      '放置位置：模块中
    Range("B1") = Join(Range("A1:A10").Value, "")
End Sub
```

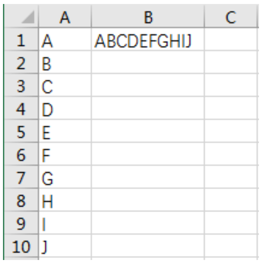
而使用 Transpose 函数后，区域的值可以合并成功。

```
Sub 合并区域的值 2()      '放置位置：模块中
   Range("B1") = Join(WorksheetFunction.Transpose(Range("A1:A10").Value), "")
End Sub
```

代码运行效果如图 10-17 所示。

在本章 10.2.6 和 10.2.7 两个小节中将会详细讲述 Join 和 Filter 函数的功能与语法。

注意：在学习工作表函数时，引用单列或者单行从而生成的值都是一维数组，而在 VBA 中引用单行或者单列生成的数组都是二维数组，可以使用 Transpose 函数将它们转换成一维数组。



	A	B	C
1	A	ABCDEFGHIJ	
2	B		
3	C		
4	D		
5	E		
6	F		
7	G		
8	H		
9	I		
10	J		

图 10-17　将区域的值合并到单元格中

Transpose 函数也有其局限性——参数的元素个数不超过 65536 个才能转置，否则会产生“类型不匹配”错误。所以当确定数组中的元素个数超过 65536 个时，应采用循环语句逐个赋值的方式转置方向。以下过程为参考标准。

```
Sub 转置超过 65536 个元素的数组()            '放置位置：模块中
    Dim Arr1, Arr2(1 To 65537)             '声明两个变量，其中 Arr2 是一维横向数组
    Arr = Range("A1:A65537").Value         '对变量 Arr 赋值，Arr 变成一维纵向数组
    For i = 1 To 65537                     '从 1 到 65537
        Arr2(i) = Arr(i, 1)                '将数组 Arr1 中的值逐个输出到 Arr2 中
    Next
End Sub
```

如果数组中有小于或等于 65536 个元素，那么直接用 Transpose 函数进行转置即可，代码如下。

```
Sub 转置不超过 65536 个元素的数组()          '放置位置：模块中
    Dim Arr1, Arr2
    Arr1 = Range("A1:A65536").Value
    Arr2 = WorksheetFunction.Transpose(Arr1)
End Sub
```

10.2.5　获取数组的上标与下标

根据数组的上标、下标，可以知道数组中的元素个数。在对数组执行循环时也需要调用数组的下标与上标。

VBA 提供了 UBound 和 LBound 函数，分别用于获取数组的上标和下标。

```
Sub test() '放置位置：模块中
    Dim Arr
    Arr = Array(1, 2, 3, 4, 5, 6)            '
    MsgBox "上标为：" & UBound(Arr) & Chr(10) & "下标为：" & LBound(Arr)
End Sub
```

此过程中使用 Array 函数创建了一个一维数组，图 10-18 表示通过 UBound 和 LBound 函数获得该数组的上标与下标。

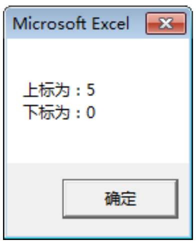


图 10-18　获取数组的上标与下标

Array 函数创建的数组默认下标为 0，而 Evaluate 函数创建的数组默认下标为 1，这种差异往往给初学者带来麻烦。VBA 提供了修改默认下标的方法——Option Base 语句。

Option Base 语句的语法如下。

```
Option Base {0 | 1}
```

即可选项包含“Option Base 0”和“Option Base 1”两项。

该语句默认的数组下标是 0，所以在任何情况下都不需要使用“Option Base 0”。当需要统一将数组的下标修改为 1 时，在模块顶部输入“Option Base 1”即可。

图 10-19 中的代码声明了一个变体型变量 Arr 和一个数组变量 Arr3，由于在模块顶部使用了“Option Base 1”，所以通过 LBound 函数获取两个数组的下标时，都能得到数值 1。

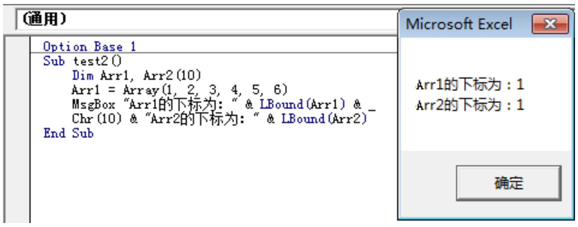


图 10–19　通过 Option Bas 语句将数组的默认下标修改为 1

当然，Ubound 和 LBound 函数并非仅限于计算数组第一维的上标和下标，它们还可以通过第二参数指定维数，从而计算任意维的上标与下标。

以下过程将分别得到一、二、三维的上标，分别为 1、5 和 20。

```
Sub 分别计算一二三维的上标()                    '放置位置：模块中
  Dim Arr(1, 5, 20)
  MsgBox UBound(Arr, 1) & Chr(10) & UBound(Arr, 2) & Chr(10) & UBound(Arr, 3)
End Sub
```

10.2.6　转换文本与数组

Split 函数可以将字符串转换成数组，而 Join 函数可以数组转换成字符串。

1. Split 函数

Split 函数可以将字符串按指定的分隔符转换成下标为 0 的一维数组，语法如下。

```
Split(expression[, delimiter[, limit[, compare]]])
```

其中 4 个参数的含义如表 10-1 所示。

表 10-1　Split函数的参数说明

参　　数	是否必选	描　　述
expression	必选参数	包含子字符串和分隔符的字符串表达式。如果是一个长度为零的字符串（""），Split则返回一个空数组
delimiter	可选参数	用于标识子字符串边界的字符串字符。如果忽略，则使用空格字符（""）作为分隔符，返回的数组仅包含expression一个元素
limit	可选参数	要返回的子字符串数量，在使用–1或者忽略参数时表示返回所有子字符串
compare	可选参数	数字值，表示判别子字符串时使用的比较方式。有四种比较方式，其中vbTextCompare表示执行文字比较，不区分大小写；vbBinaryCompare表示执行二进制比较，要区分大小写。在忽略参数时表示采用vbTextCompare

通常在字符串中有明显的分隔符时才有必要使用 Split 函数，例如将“1，2，3”以逗号为分

隔符转换成包含 3 个元素的数组，再如将“湖南-张家界-天门山-天门洞”以“-”为分隔符转换成包含 4 个元素的数组……

以下过程可以展示 Split 函数的工作模式，以及它生成的数组的特点。

```
Sub Split 函数的应用()                                  '放置位置：模块中
    Dim i As Byte, Arr As Variant                       '声明变量
    Arr = Split("湖南-张家界-天门山-天门洞", "-")        '将字符串以“-”为分隔符转换成一维数组
  '计算数组的上标（由于下标是 0，所以上标等于数组元素个数–1）
   i = UBound(Arr)
  '将数组的值导出到区域中，其中 Resize 的第二参数必须是数组的上标+1
    Range("A1").Resize(1, i + 1) = Arr
End Sub
```

代码运行效果如图 10-20 所示。

	A	B	C	D	E
1	湖南	张家界	天门山	天门洞	
2					

图 10–20　将字符串转换成数组并导出到区域中

由于 Split 函数生成的数组总是下标为 0，不受 Option Base 语句的影响，因此它所生成的数组的元素个数一定等于数组的上标加 1。所以在以上过程中 Range.Resize 属性的第二参数使用了“i+1”。

本例案例文件请参考：..\ 第 10 章\ 10-4 Split 函数的应用.xlsm

其实，Split 函数最常见的应用是根据路径获取根目录或者文件名称。

例如某文件的路径是“D:\5 月\生产表\总表.xlsx”，要求计算该文件名称和所在的根目录，采用数组函数 Split 和 UBound 可以处理此问题，代码如下。

```
Sub  从路径中获取文件名称与根目录()          '放置位置：模块中
    Dim Mystr As String                     '声明变量
    Mystr = "D:\5 月\生产表\总表.xlsx"        '对变量赋值为文件路径
  '从路径中取文件名与根目录名称
   MsgBox Split(Mystr, "\")(UBound(Split(Mystr, "\")))   & Chr(10) & Split(Mystr, "\")(0)
End Sub
```

在过程中 Split 函数对路径以“\”为分隔符，转换成一个一维数组，然后利用 UBound 函数计算该数组的上标，以上标作为数组的索引号可以提取数组中最后一个元素的值，即文件名称。而在以 0 作为数组的索引号时，则可以提取数组中第一个元素的值，即路径的根目录名称，如图 10-21 所示。

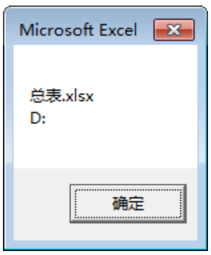


图 10–21　文件名称与根目录

当然也可以用 Split 函数获得文件的后缀名，获取思路与获取文件名称一致，代码如下。

```
Sub 从路径中获取文件的后缀名()
    Dim Mystr As String                          '声明变量
    Mystr = "D:\5 月\生产表\总表.xlsx"            '对变量赋值为文件路径
    MsgBox Split(Mystr, ".")(UBound(Split(Mystr, ".")))   '从路径中取文件后缀名
End Sub
```

本例案例文件请参考：..\ 第 10 章\ 10-5 从路径中取文件名与根目录.xlsm

2．Join 函数

Join 函数可用于连接数组中的所有元素，从而创建一个新的字符串，可随意指定分隔符，其语法如下。

```
Join(sourcearray[, delimiter])
```

其中参数 sourcearray 代表数组，参数 delimiter 代表分隔符。如果忽略分隔符，则默认采用空格作为分隔符。

Join(Array("D:", "生产表", "5 月"), "\")——转换结果为“D:\生产表\5 月”。

Join(Array("2013", "09", "28"), "-")——转换结果为“2013-9-28”。

Join(Array("VBA", "果然", "神奇"), "")——转换结果为“VBA 果然神奇”。

可以将 Join 函数理解为合并数组的值，并在适当的位置添加分隔符。

事实上也可以使用 Join 函数合并区域的值，不过必须配合 Transpose 函数来实现，在本章 10.2.4 节中有此类案例应用。

10.2.7　筛选数组

Filter 函数用于返回一个下标从 0 开始的数组，该数组包含基于指定筛选条件的一个字符串数组的子集。通俗地讲，Filter 函数就是对一个一维数组进行筛选，从而产生一个新的数组。其语法如下：

```
Filter(sourcesrray, match[, include[, compare]])
```

其中各参数的含义如表 10-2 所示。

表 10-2　Filter函数的参数说明

参数	含　义
Sourcearray	必选参数，要执行搜索的一维字符串数组
Match	必选参数，要搜索的字符串
Include	可选参数，Boolean值，表示新数组是否包含match字符串。如果值为True，就返回包含match子字符串的数组子集，否则返回不包含match子字符串的数组子集
Compare	可选参数，数字值表示所使用的字符串比较类型，通常用于控制是否区分大小写

Filter 函数的第三参数比较重要，它决定筛选时采用包含或者不包含字符串的方式。

Filter 函数只能筛选一维数组，对二维、三维数组无效。

以下案例对数组执行了两种筛选方式，生成两个新数组，然后用 Join 函数合并数组中的每个元素，并显示在提示框中，如图 10-22 所示。

```
Sub 找出姓罗与不姓罗的所有姓名()   '放置位置：模块中
    Dim Arr1, Arr2, Arr3
    Arr1 = Array("罗成", "骆宾王", "柳如是", "曲非烟", "罗贯中", "罗通")
    Arr2 = Filter(Arr1, "罗", True)
    Arr3 = Filter(Arr1, "罗", False)
    MsgBox "所有姓罗的姓名:" & Chr(10) & Join(Arr2, ",") & Chr(10) & _
```

```
        "所有不姓罗的姓名：" & Chr(10) & Join(Arr3, ",")
End Sub
```

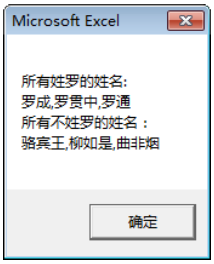


图 10-22　两种数组筛选方式结果

除了从数组中筛选出子集生成一个新数组，也常使用 Filter 函数判断一个数组中是否包含某个元素。例如判断数组中是否包含“曲非烟”，那么可用以下代码：

```
Sub 判断是否包含曲非烟()                                              '放置位置：模块中
    Dim Arr1                                                          '声明变量
    Arr1 = Array("罗成", "骆宾王", "柳如是", "曲非烟", "罗贯中", "罗通")   '对变量赋值
  '根据筛选结果判断是否包含“曲非烟”
   MsgBox IIf(UBound(Filter(Arr1, "曲非烟", True)) < 0, "不包含", "包含")
End Sub
```

过程的执行结果为“不包含”，过程的原理如下。

Filter 函数的筛选结果是一维数组，且下标为 0，上标大于等于 0，如果上标小于 0，那么说明未成功生成数组，即不包含筛选条件。所以本例利用 UBound 函数计算 Filter 函数生成数组的上标，并配合 IIf 函数即可达成需求。

10.3　案例分析

VBA 在数组方面的理论内容不多，掌握数组的应用也较简单。

在实际工作中通常使用数组替代区域，从而提升数据的读写速度。本节通过 7 个案例展示数组的应用，从而加深读者对数组的理解。

10.3.1　将指定区域的单词统一为首字母大写

【案例要求】将指定区域中的英文单词全部修改为首字母大写。

【知识要点】Application.InputBox、Intersect、UBound 和 StrConv。

【程序代码】

```
Sub 将选区的单词转换成首字母大写()                            '放置位置：模块中
    Dim Arr, Rng as range                                     '声明变量
    On Error Resume Next                                      '当程序出错时继续执行下一行
    Set Rng = Application.InputBox("请选择操作区域", "确定区域", , , , , , 8)  '选择待转换的区域
   '如果 Rng 变量赋值（单击了“取消”按钮）那么结束过程
    If Rng Is Nothing Then Exit Sub
   '将选择的区域与活动工作表已用区域的交集赋予变量
     Arr = Intersect(ActiveSheet.UsedRange, Rng).Value
     Dim  RowCount As Integer, ColumnCount As Integer         '声明变量
     For RowCount = 1 To UBound(Arr)                          '遍历数组的每一行
     For ColumnCount = 1 To UBound(Arr, 2)                    '遍历数组的每一列
```

```
            '将数组中的元素转换为首字母大写
              Arr(RowCount, ColumnCount) = StrConv(Arr(RowCount, ColumnCount),
    vbProperCase)
            Next ColumnCount
        Next RowCount
        Intersect(ActiveSheet.UsedRange, Rng) = Arr              '将数组的值写回区域中
    End Sub
```

在执行以上过程时将弹出“确定区域”的输入框，此时用鼠标光标选择 A1:5C 区域即可，输入框中将自动产生选区的地址，如图 10-23 所示。当单击输入框中的“确定”按钮后，程序瞬间将选区中的所有单词转换成首字母大写，执行结果如图 10-24 所示。

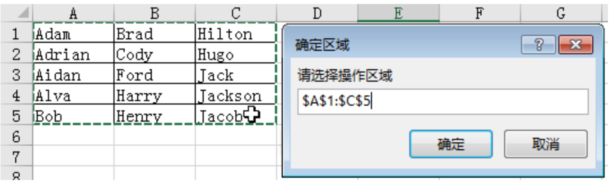


图 10-23　选择区域

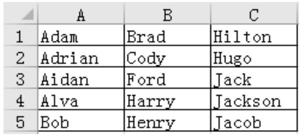


图 10-24　执行结果

【代码分析】

1．Application.InputBox 的最后一个参数必须使用 8，才能实现鼠标光标单击区域时自动在输入框中产生地址且返回 Range 对象。同时，由于 Application.InputBox 的返回值是 Range 对象，所以对 Rng 变量赋值时要采用 Set 语句。

2．在 Application.InputBox 的输入框中按下“取消”按钮后，Application.InputBox 的返回值为 False，将 Flase 赋值给变量 Rng 会出错，因此需要在此代码前添加“On Error Resume Next”从而屏蔽错误提示信息。

3．尽管工作表中需要转换的区域是 A1:C5，但由于程序中使用了 Intersect 方法获取 Rng 与活动工作表的已用数据区域的交集，所以选择区域时可以更自由一些，直接选择 A:C 区域也可以，程序会自动忽略空白区域，仅转换选区与已用数据区域的交集。

4．本例中对变量 Arr 一次性赋值，从而生成数组，所以在定义变量时一定要采用变体型，不能在变量名称后面指定数组维数。

5．数组变量中每一个元素都允许随意写入、提取或者修改，所以在本例中先将区域的值赋予变量，然后在双循环语句中逐一修改数组每个元素的值，将它们转换成首字母大写状态。待所有字符转换成功后一次性导入到区域。

6．StrConv 函数是一个功能强大的转换函数，其第二参数在使用 vbUpperCase 和 vbLowerCase、vbProperCase 时，可分别实现将英文单词转换成全部大写、全部小写或首字母大写的形式。

7．对于需要大量读取和写入单元格的操作，都极有必要使用数组，从而大幅提升工作效率。

本例案例文件请参考：..\ 第 10 章\ 10-7 将指定区域的单词统一为首字母大写.xlsm

10.3.2　罗列不及格学生姓名、科目和成绩

【案例要求】在图 10-25 中包含若干学生的成绩，要求将所有不及格学生的姓名、科目与成绩一并罗列出来，排为三列存放在 J 列到 L 列中。

【知识要点】ReDim Preserve、UBound、Transpose、Resize 和 Array。

【程序代码】

```
Sub 罗列不及格学生信息()     '放置位置：模块中
    Dim Arr, Arr2(), i As Integer, j As Integer, TargetCount As Integer '声明变量
    '将活动工作表中已用数据区域的值赋予变量
    Arr = ActiveSheet.UsedRange.Value
    '遍历数组的每一行（标题列例外）
    For i = 2 To UBound(Arr)
      '遍历数组的每一列（标题列例外）
      For j = 2 To UBound(Arr, 2)
            '如果成绩值小于 60
            If Arr(i, j) < 60 Then
                '累加计数器，此变量的值等于符合条件的成绩个数
                 TargetCount = TargetCount + 1
                '重置数组变量的维数和上标（维数只能重置一次，最末一维上标不限重置次数）
                ReDim Preserve Arr2(1 To 3, 1 To TargetCount)
                '将符合条件的姓名赋值到数组 Arr2 的第一行中
                Arr2(1, TargetCount) = Arr(i, 1)
              '将符合条件的科目赋值到数组 Arr2 的第二行中
            Arr2(2, TargetCount) = Arr(1, j)
            '将符合条件的成绩赋值到数组 Arr2 的第三行中
            Arr2(3, TargetCount) = Arr(i, j)
        End If
      Next j
   Next i
   '一次性将标题写入到 J1:L1 区域中
   Range("J1:L1") = Array("姓名", "科目", "成绩")
  '将数组的值导出到 J2 开始的区域
    Range("J2").Resize(TargetCount, 3) = WorksheetFunction.Transpose(Arr2)
    '对 J1 单元格的当前区域添加边框
    Range("J1").CurrentRegion.Borders.LineStyle = xlContinuous
End Sub
```

当执行过程后将得到如图 10-26 所示的结果。

	A	B	C	D	E	F	G	H
1	姓名	语文	数学	化学	政治	计算机	英语	法律
2	计尚云	76	89	52	80	73	65	81
3	赵国	83	63	64	92	92	80	100
4	罗至贵	96	61	85	99	62	77	55
5	徐大鹏	100	84	50	79	55	55	90
6	张志坚	64	52	65	69	65	98	99
7	朱千文	70	64	58	58	82	70	71
8	赵秀文	86	66	82	60	59	79	54
9	梁爱国	73	96	63	90	69	64	96
10	梁兴	82	82	71	54	78	85	96
11	陈随机	92	51	77	96	71	84	75

图 10-25　成绩表

	G	H	I	J	K	L
1	英语	法律		姓名	科目	成绩
2	65	81		计尚云	化学	52
3	80	100		罗至贵	法律	55
4	77	55		徐大鹏	化学	50
5	55	90		徐大鹏	计算机	55
6	98	99		徐大鹏	英语	55
7	70	71		张志坚	数学	52
8	79	54		朱千文	化学	58
9	64	96		朱千文	政治	58
10	85	96		赵秀文	计算机	59
11	84	75		赵秀文	法律	54

图 10-26　不及格学生信息

【代码分析】

1．由于无法获知符合条件的成绩个数，所以声明数组变量 Arr2 时不指明数组的维数和每一维的下标、上标，而是在循环语句中通过 ReDim Preserve 语句为数组重置维数和上标。不过需要声明的是，ReDim Preserve 语句并不可以随意修改数组的维数和下标、上标，它只能修改一次数组的维数和每一维的下标，而最末一维的上标则可以反复地修改。

2．本例中所有符合条件的数据所形成的数组包含 3 列，即姓名、科目和成绩，其行数则由符合条件的成绩个数决定。换言之，数组的第二维的下标和上标可以固定为 1 和 3，但第一维的

下标不确定，而 VBA 规定 ReDim Preserve 语句只能反复修改最末一维的下标，所以指定变量的维数时不能使用“ReDim Preserve Arr2(1 To TargetCount，1 To 3)”，而是使用“ReDim Preserve Arr2(1 To 3, 1 To TargetCount)”，当对数组赋值完成后再使用转置函数 Transpose 转置方向。

3．本例中对变量 Arr2 赋值的方式比较重要，代码如下：

```
Arr2(1, TargetCount) = Arr(i, 1)
Arr2(2, TargetCount) = Arr(1, j)
Arr2(3, TargetCount) = Arr(i, j)
```

由于数组 Arr2 包含三行，列数由变量 TargetCount 决定，即 TargetCount 列是最后一列，所以将符合条件的姓名、科目和成绩追加到数组中时分别采用 Arr2(1, TargetCount)、Arr2(2, TargetCount)和 Arr2(3, TargetCount)。

而 Arr(i, 1)的来历是由于姓名处于数组 Arr 的第 1 列第 *i* 行；科目处于数组 Arr 的第 1 行第 *j* 列，所以引用科目时使用(1, j)；成绩处于第 *i* 行第 *j* 列，所以引用成绩时采用 Arr(i, j)。

4．将数组 Arr2 导入工作表时，不能像 Range.Copy 那样只指定目标区域左上角单元格，而是必须指定与数组相同高度与宽度的区域，所以本例对 Range("J2")单元格的 Resize 属性指定高度为 TargetCount，宽度为 3。其中 TargetCount 表示符合条件的成绩个数，3 代表姓名、科目和成绩三项标题所占用的列宽。

如果本例不使用数组，那么代码执行时间将延长到两倍以上。

本例案例文件请参考：..\ 第 10 章\ 10-8 罗列不及格学生姓名、科目和成绩.xlsm

10.3.3 将字符串合并到区域

【案例要求】对选区批量添加指定的字符串，并由用户决定放置在原文之前还是原文之后。

【知识要点】TypeName、Intersect、UBound、IF Then、For Next 和 Replace。

【程序代码】

```
Sub 将字符串合并到区域()    '放置位置：模块中
  Dim  Rng As Range'声明变量
   '如果选区不是单元格，那么提示用户，然后结束过程
   If TypeName(Selection) <> "Range" Then MsgBox "请选择单元格！", 65, "友情提示": Exit Sub
   '将选区与活动工作表的已用数据区域赋值给变量 Rng
   Set Rng = Intersect(Selection, ActiveSheet.UsedRange)
   '如果 Rng 未初始化（选择与已用数据区域不重叠）
   If Rng Is Nothing Then
     '那么提示用户，然后结束过程
     MsgBox "不要选择空白区域", vbInformation, "友情提示"
     Exit Sub
Else '否则
    '声明变量
     Dim Msg As VbMsgBoxResult, ResultStr, Arr, RowCount As Integer, ColumnCount
As Integer
    '弹出一个输入框让用户指定需要合并到选区中的字符串，可以手动输入字符，也可以选择单元格
     ResultStr = Application.InputBox("请选择待合并的对象,一个单元格即可" & Chr(10)
& "程序将会把该单元格的值追加为选区的前缀或者后缀。", "待合并对象", "A1", , , , , 2)
     If ResultStr = "False" Then Exit Sub     '如果单击了“取消”按钮，那么结束过程
     '弹出信息框，让用户预览两种合并效果，同时让用户从中选择一种
     Msg = MsgBox("选择是:合并到原数据之前        预览:" & ResultStr & Replace(Selection(1), Chr(10), "")
& Chr(10) & Chr(10) & "选择否:合并到原数据之后        预览:" & Replace(Selection(1), Chr(10), "") & ResultStr,
```

```
vbYesNo, "是否合并到前缀? ")
        Arr = Rng.Value                         '将区域 Rng 的值赋予变量 Arr
        For RowCount = 1 To UBound(Arr)         '遍历数组的每一行
            For ColumnCount = 1 To UBound(Arr, 2) '遍历数组的每一列
                If Msg = vbYes Then             '如果用户单击了“是”按钮
                    '如果数组中第 RowCount 行第 ColumnCount 列的长度大于 0
                    If Len(Arr(RowCount, ColumnCount)) > 0 Then Arr(RowCount,
    ColumnCount) = ResultStr & Arr(RowCount, ColumnCount)
                Else    '否则（即选择了“否”按钮）
                    '如果数组中第 RowCount 行第 ColumnCount 列的长度大于 0
                    If Len(Arr(RowCount, ColumnCount)) > 0 Then Arr(RowCount,
    ColumnCount) = Arr(RowCount, ColumnCount) & ResultStr
                End If
            Next
        Next
        Rng = Arr    '修改完成后，将数组 Arr 的值一次性写入 Rng 区域中
    End If
End Sub
```

在执行以上过程前需要选择一个数据区域，如果选择空白区域再执行，那么将自动结束过程。

如果在选择非空区域后执行过程，那么将弹出如图 10-27 所示的输入框，可在其中输入任意需要合并到区域中的字符串。当输入字符“广东”并单击“确定”按钮后，会弹出如图 10-28 所示的预览窗口。在预览窗口中包含了合并到原数据之前与之后的两种效果，可以根据预览效果选择合并方式。

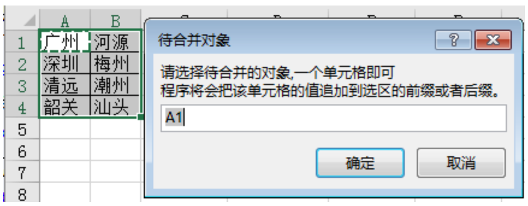


图 10-27　指定合并字符

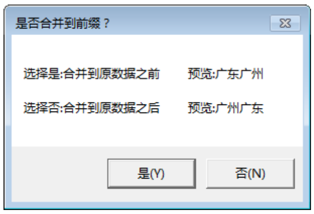
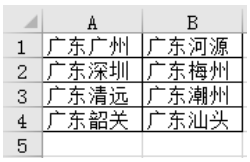


图 10-28　预览合并效果

假设单击“是”按钮，那么合并后的效果如图 10-29 所示。



	A	B
1	广东广州	广东河源
2	广东深圳	广东梅州
3	广东清远	广东潮州
4	广东韶关	广东汕头
5		

图 10-29　合并结果

事实上，在输入框中也允许使用点选的方式引用单元格，即在使用鼠标光标选择单元格时自动在输入框中产生引用区域的地址。不过在合并时仅采用区域左上角的单元格的值，而不是将所选区中的字符都参与合并。假设在弹出如图 10-30 的输入出框后选择了 D1 单元格，并且在接下来弹出的对话框中单击了“否”按钮，那么合并结果如图 10-31 所示。

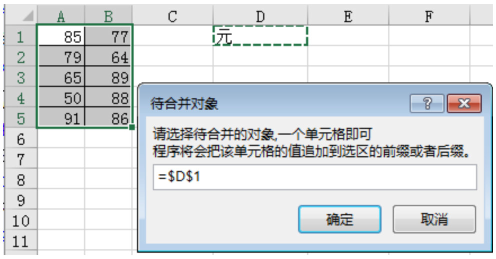


图 10-30　支持单元格引用

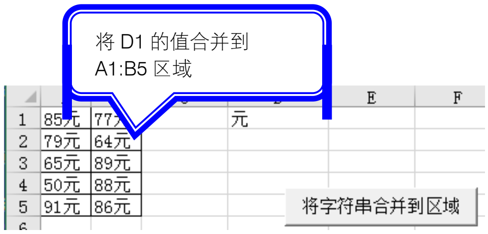


图 10-31　合并结果

【代码分析】

1．由于程序的功能是将字符串合并到选区中每个非空单元格之前或之后，所以在执行合并之前需要确保当前选择的对象是单元格，而且是非空单元格。所以在过程的前面需要采用 If 语句执行两次判断，当不符合条件时自动结束过程，这是完善程序的必要手段。

2．Application.InputBox 方法创建的输入框既可以手动输入字符串，也可以通过选择单元格的方式产生单元格引用。不过当它的第八参数不使用 8 时，Application.InputBox 方法无法返回 Range 对象。

3．MsgBox 函数有返回值，如果需要将此返回值赋予变量，那么在声明此变量时应使用 VbMsgBoxResult 型。不过，在工作中 VbMsgBoxResult 型的数据类型用得较少，不利于记忆，所以也可以在声明变量时采用变体型，不会对执行程序有任何影响，只是在输入代码时不提供即时信息。

当变量声明为 VbMsgBoxResult 型后，在对变量赋值时会产生如图 10-31 所示的快速信息。

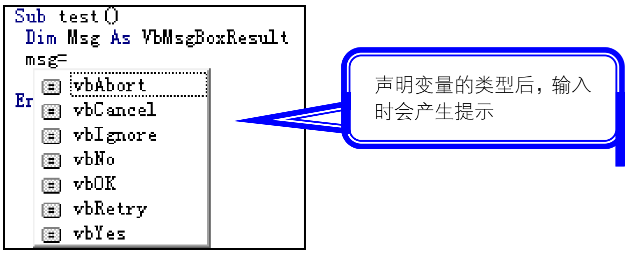


图 10-32　VbMsgBoxResult 型变量的快速信息

4．过程中的代码“Replace(Selection(1), Chr(10), "")”用于去除选区每一个单元格中的换行符，从而使其显示在信息框时更美观。Selection(1)表示选区左上角的单元格，此处必须使用索引号 1，否则在选区包含多个单元格时将导致程序出错。

5．由于本例可以直接修改数组 Arr 中的值，然后将数组导出到区域中，因此只采用一个数组。在数组的应用案例中，更多情况下是使用两个数组变量，一个用于读取数据，一个用于写入数据。

本例案例文件请参考：..\ 第 10 章\ 10-9 合并字符串到选区.xlsm

10.3.4　将职员表按学历拆分成多个工作表

【案例要求】将图 10-33 中的职工信息表按学历拆分成多个工作表，每个工作表的名字以学历命名。学历包含小学、初中、中专和大学四类。

	A	B	C	D	E
1	姓名	学历	年龄	性别	
2	计尚云	大学	80	男	
3	赵国	中专	100	男	
4	罗至贵	中专	96	男	
5	徐大鹏	中专	61	男	
6	张志坚	大学	85	男	
7	朱千文	大学	99	女	
8	赵秀文	初中	62	女	
9	梁爱国	大学	77	男	
10	梁兴	中专	55	男	
11	陈随机	大学	100	男	

图 10–33　职工信息表

【知识要点】Array、UBound、ReDim Preserve、Transpose、Erase 和 CurrentRegion。

【程序代码】

```
    Sub 将职员表按学历拆分成多个工作表()                      '放置位置：模块中
        Dim 学历, data, Arr()                                   '声明变量
        Dim i As Byte, RowCount As Integer, TargetCount As Integer
        On Error Resume Next                                    '代码错误时继续执行下一步
        学历 = Array("小学", "初中", "中专", "大学")            '将四种学历组成数组赋值给变体型变量
        data = Range("a1").CurrentRegion.Value
        Application.DisplayAlerts = False                       '关闭提示（删除工作表时会有提示）
'将 A1 单元格的当前区域赋予变量 data
        For i = 0 To UBound(学历)                               '遍历“学历”数组的每一个元素
            '将变量 TargetCount 指定为 0（每一轮循环前必须以 0 开始，否则计数不准确）
             TargetCount = 0
             Worksheets(学历(i)).Delete                          '删除以学历名称命名的工作表
            '新建一个工作表，以学历命名
             Worksheets.Add(after:=Worksheets(Worksheets.Count)).Name = 学历(i)
            '遍历数组 data 的每一行（标题行例外，所以从 2 开始）
             For RowCount = 2 To UBound(data)
            '如果 data 的第二列的值等于数组“学历”第 i 个元素
             If data(RowCount, 2) = 学历(i) Then
                  '累加计数器，表示本轮循环中有多少个学历与“学历(i)”相同
                   TargetCount = TargetCount + 1
                '重置数组 Arr 的维数、下标与上标（其中最末一维的上标根据实际情况随时增加）
                 ReDim Preserve Arr(1 To 4, 1 To TargetCount)
                '将数组 data 中的姓名追加到数组 Arr 的第一行第 TargetCount 列中
                 Arr(1, TargetCount) = data(RowCount, 1)
                '将数组 data 中的学历追加到数组 Arr 的第二行第 TargetCount 列中
                 Arr(2, TargetCount) = data(RowCount, 2)
                '将数组 data 中的年龄追加到数组 Arr 的第三行第 TargetCount 列中
                 Arr(3, TargetCount) = data(RowCount, 3)
                '将数组 data 中的性别追加到数组 Arr 的第四行第 TargetCount 列中
                 Arr(4, TargetCount) = data(RowCount, 4)
             End If
          Next RowCount
            '如果 TargetCount 的值大于 0（表示找到了符合条件的职工信息）
             If TargetCount > 0 Then
            '引用数组“学历”中第 i 个元素命名的工作表
             With Worksheets(学历(i))
               .Range("A1:D1") = Array("姓名", "学历", "年龄", "性别") '写入标题
               '将数组转置后导入工作表
               .Range("A2").Resize(TargetCount, 4) = WorksheetFunction. Transpose(Arr)
```

```
                .UsedRange.Borders.LineStyle = xlContinuous
            End With
           '清除数组中的值，进入下一轮循环（否则会保留前面的数据，从而导致拆分出错）
             Erase Arr
           End If
       Next i
   End Sub
```

执行以上过程后将会把职工信息表拆分成“小学”、“初中”、“中专”和“大专”四个工作表，每个表中保存同一类学历的职工信息，结果如图 10-34 所示。



	A	B	C	D	E	F
1	姓名	学历	年龄	性别		
2	计尚云	大学	80	男		
3	张志坚	大学	85	男		
4	朱千文	大学	99	女		
5	梁爱国	大学	77	男		
6	陈随机	大学	100	男		
7	刘子中	大学	84	男		
8	诸有光	大学	50	男		
9	李文新	大学	55	女		
10	梁今明	大学	65	男		

图 10-34　拆分结果

【代码分析】

1．在本例中使用了两个数组变量，第一个用于一次性读取职工信息表中的数据，所以在声明变量时采用变体型变量；第二个数组用于逐个写入符合条件的职工信息，所以在声明变量时采用数组变量但不指定数组维数，当进入 For Next 循环语句后利用 ReDim Preserve 语句重置数组的维数和上标。不过由于 ReDim Preserve 只能更改数组最末维的上标，所以在定义数组的维数时不能定义为列数等于 4、行数等于符合条件的数据个数，而是定义为行数等于 4、列数等于符合条件的数据个数，当循环完成后再利用 Transpose 函数将数组重置方向并导出工作表。

2．数组 Arr 需要多次使用，所以每一次使用完后都需要通过 Erase 语句释放变量的值，避免影响下一轮赋值。

3．将数组 data 的值赋予数组 Arr 时，由于两者的方向相反，所以索引号的顺序也相反，代码如下。

```
   Arr(1, TargetCount) = data(RowCount, 1)
   Arr(2, TargetCount) = data(RowCount, 2)
   Arr(3, TargetCount) = data(RowCount, 3)
   Arr(4, TargetCount) = data(RowCount, 4)
```

4．Erase 语句在释放数组时会将数组的值全部清空，同时将数组的维数、上标等一切信息都清除，所以在下次引用该动态数组之前必须使用 ReDim 语句重新定义该数组变量的维数，否则无法对该数组写入新值。

5．在使用代码删除工作表时，不管工作表是否空白都会弹出对话框，从而影响操作。因此在使用 Worksheets.Delete 方法之前必须使用“Application.DisplayAlerts = False”语句关闭提示。要注意的是，Worksheets.Delete 方法处于循环语句之中，而“Application.DisplayAlerts = False”语句不适宜放在循环语句中，否则该语句将反复执行，浪费资源。

本例案例文件请参考：..\ 第 10 章\ 10-10 将职员表整理为学历分类表.xlsm

10.3.5　将选区的数据在文本与数值间互换

【案例要求】将选区的数据在文本与数值间互换，由用户选择转换方式和区域。

【知识要点】UBound、Intersect、IF…Then…End If、For Next 和 NumberFormatLocal。

【程序代码】

```
Sub 文本与数值互换() '放置位置：模块中
  Dim 转换方式 As VbMsgBoxResult, Arr(), row As Long, column As Long '定义变量 '让用户选择转换方式
  转换方式 = MsgBox("选择是：将文本转换成数字；" + Chr(10) + "选择否：将数字转换成文本。", vbYesNo, "操作方式")
    With Intersect(Selection, ActiveSheet.UsedRange)         '引用选区与已用数据区域的交集
    If 转换方式 = vbYes Then                                  '如果选择将文本转换成数字
       .NumberFormatLocal = "G/通用格式"                      '修改数字格式为 G/通用格式
       .Value = .Value                                        '转换成值
    Else                                                      '否则
       Arr = .Value                                           '将选区与已用区域的交集写入数组
       For row = 1 To UBound(Arr)                             '逐行循环
         For column = 1 To UBound(Arr, 2)                     '逐列循环
             '如果是数字则前置“'”使其成为文本
              If Len(Arr(row, column)) > 0 Then Arr(row, column) = "'" & Arr(row, column)
         Next
       Next
       .Value = Arr                                           '将数组一次性写入区域中
    End If
  End With
End Sub
```

在图 10-35 中的成绩区域有部分单元格是数值格式，有部分单元格是文本格式，所以 B12 单元格的公式“=AVERAGE(B2:B11)”计算结果不可能准确。对于这种情况有必要通过程序使它们格式统一。

选择 B2:D11 区域后，执行过程“文本与数值互换”将弹出如图 10-36 所示的提示框。

	A	B	C	D
1	姓名	语文	数学	英语
2	计尚云	52	55	82
3	赵国	71	87	42
4	罗至贵	93	67	49
5	徐大鹏	90	62	86
6	张志坚	69	45	65
7	朱千文	99	99	87
8	赵秀文	94	82	88
9	梁爱国	52	80	59
10	梁兴	98	62	81
11	陈随机	68	93	97
12	平均	74.875	74.44444	77.11111

图 10–35　成绩区域有文本导致计算不正确

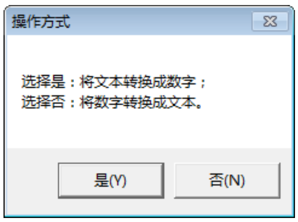
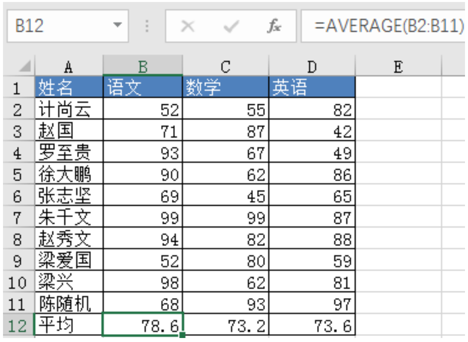


图 10–36　让用户选择转换方式

单击提示框中的“是”按钮将把选区中的所有单元格统一为数值格式，这样公式“=AVERAGE(B2:B11)”才能获得正确结果，效果如图 10-37 所示。

如果需要将单元格批量修改为文本格式，执行本例的过程后，在如图 10-36 所示的提示框中单击“否”按钮即可。



B12 =AVERAGE(B2:B11)

	A	B	C	D	E
1	姓名	语文	数学	英语	
2	计尚云	52	55	82	
3	赵国	71	87	42	
4	罗至贵	93	67	49	
5	徐大鹏	90	62	86	
6	张志坚	69	45	65	
7	朱千文	99	99	87	
8	赵秀文	94	82	88	
9	梁爱国	52	80	59	
10	梁兴	98	62	81	
11	陈随机	68	93	97	
12	平均	78.6	73.2	73.6	

图 10–37　文本转数值后的公式计算结果

【代码分析】

1．让文本格式的数字显示为数值，需要先将单元格的数字格式修改为“G/通用格式”，然后重新对单元格赋值，该操作可以批量完成。

如果要让数值显示为文本，则在数字前添加半角状态的单引号即可，不过该方法无法批量操作，所以本例将选区的值导入到数组中，然后在数组的所有非空元素前添加“'”，最后一次性将数组的值导入区域。

2．Range. NumberFormatLocal 属性表示单元格的数字格式，相当于“设置单元格格式”对话框中“数字”选项卡中的各项设置功能。可以通过该对话框获取各种数字格式，然后应用于 VBA 中。图 10-38 显示了 Excel 自带的单元格数字格式代码，这些代码都可以直接用于 VBA，对 Range. NumberFormatLocal 属性赋值，从而调整单元格的显示状态。

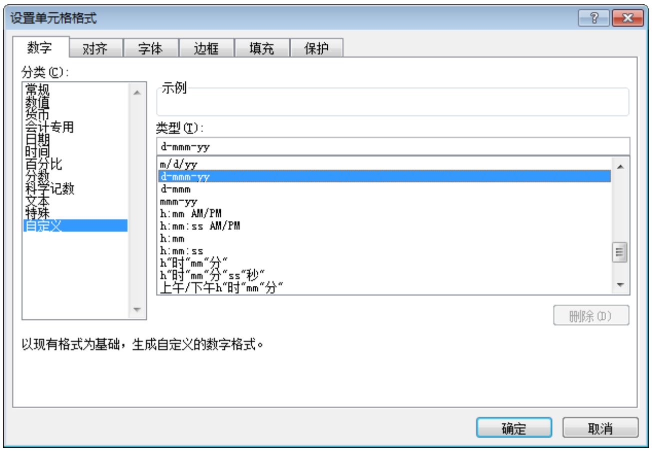


图 10–38　Excel 自带的数字格式代码

本例案例文件请参考：..\ 第 10 章\ 10-11 将选区的数据在文本与数值间互换.xlsm

10.3.6　获取两列数据的相同项

【案例要求】图 10-39 中 A、B 两列包含两届参赛队员的信息表，要求比较两列数据的异同，并将其分别罗列出来，保存在 D 列和 E 列中。



	A	B	C
1	2011届	2012届	
2	计尚云	朱千文	
3	赵国	诸有光	
4	罗至贵	周华章	
5	徐大鹏	曲华国	
6	张志坚	李文新	
7	朱千文	陈随机	
8	赵秀文	陈强生	
9	梁爱国	刘文喜	
10	梁兴	赵国	
11	陈随机	柳三秀	

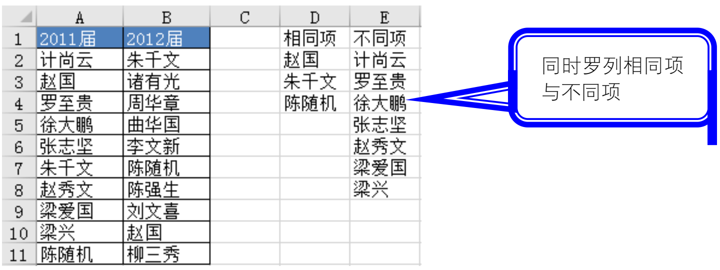
图 10-39　两届参赛队员信息表

【知识要点】Transpose、Filter、UBound、ReDim Preserve、For Next 和 Resize。

【程序代码】

```
Sub 获取两列的相同项与不同项()                              '放置位置：模块中
  Dim Arr1, Arr2, Arr3(), Arr4(), Item As Integer, Count1 As Integer, count2 As Integer '声明变量
  '分别将两个纵向的区域转换成一维数组
   Arr1 = WorksheetFunction.Transpose(Range([A2], Cells(Rows.Count, 1).End
(xlUp)).Value)
   Arr2 = WorksheetFunction.Transpose(Range([B2], Cells(Rows.Count, 2).End
(xlUp)).Value)
   For Item = 1 To UBound(Arr1)                          '遍历数组 Arr1 的每一行
   If UBound(Filter(Arr2, Arr1(Item), True)) >= 0 Then   '如果 Arr2 中包含 Arr1 中的第 Item 个元素
        Counter1 = Counter1 + 1                          '那么累加计数器（统计符合条件的数据个数）
        ReDim Preserve Arr3(1 To Counter1)               '重置一维数组的上标
        Arr3(Counter1) = Arr1(Item)                      '将数组 Arr1 中第 Item 个元素赋值给 Arr3
  Else                                                   '否则（Arr2 中不包含 Arr1 中的第 Item 个元素）
      Counter2 = Counter2 + 1                            '那么累加计数器（统计不符合条件的数据个数）
      ReDim Preserve Arr4(1 To Counter2)                 '重置一维数组的上标
      Arr4(Counter2) = Arr1(Item)                        '将数组 Arr1 中第 Item 个元素赋值给 Arr4
  End If
  Next Item
  Range("D1:E1") = Array("相同项", "不同项")              '在 D1:E1 写入标题
  Range("D2").Resize(Counter1, 1) = WorksheetFunction.Transpose(Arr3) '将相同项保存在 D 列
  Range("E2").Resize(Counter2, 1) = WorksheetFunction.Transpose(Arr4) '将不同项保存在 E 列
End Sub
```

执行以上过程后，程序会瞬间找出两届参赛队员的相同项与不同项，并将它们分别罗列在 D 列和 E 列中，如图 10-40 所示。



	A	B	C	D	E
1	2011届	2012届		相同项	不同项
2	计尚云	朱千文		赵国	计尚云
3	赵国	诸有光		朱千文	罗至贵
4	罗至贵	周华章		陈随机	徐大鹏
5	徐大鹏	曲华国			张志坚
6	张志坚	李文新			赵秀文
7	朱千文	陈随机			梁爱国
8	赵秀文	陈强生			梁兴
9	梁爱国	刘文喜			
10	梁兴	赵国			
11	陈随机	柳三秀			

图 10-40　提取两列姓名的相同项与不同项

【代码分析】

1．首先通过 Transpose 函数将两列数据转换成一维数组，并赋值给两个变体型变量，此时

两个变量被转换成数组。由于区域引用通过 Transpose 函数转换后是横向的一维数组，所以可以作为 Filter 函数的参数参与运算。

如果要求比较的对象是两行数据，那么必须使用 Transpose 函数转置两次才能将区域引用转换成一维数组，否则不支持 Filter 函数。

例如判断 A1:E1 区域是否包含“上海”二字，如果采用数组思路实现，那么应在 Range("A1:E1") 对象外套两层 Transpose，从而使一维横向的区域引用转换成一维数组，然后通过 Filter 函数以“上海”作为筛选条件生成新数组，并根据数组的上标判断在 A1:E1 区域中是否包含“上海”二字，完整代码如下。

```
Sub 判断A1到E1是否包含上海()  '放置位置：模块中
    Dim Arr
    Arr = WorksheetFunction.Transpose(WorksheetFunction.Transpose (Range ("A1:E1")))
    MsgBox IIf(UBound(Filter(Arr, "上海", True)) <= 0, "不包含", "包含")
End Sub
```

如果直接将 Range("A1:E1") 赋值给变量 Arr，或者将转置后的 WorksheetFunction. Transpose(Range("A1:E1"))赋值给 Arr，那么数组 Arr 都不能作为 Filter 函数的参数参与运算，因为这两种方式产生的数组都不是一维数组。

2．判断一个数组中是否包含某个字符也可以采用 Instr+Join 组合实现。其中 Join 函数用于将数组转换成字符串，Instr 函数则用于计算查找的目标字符在此字符串中首次出现的位置，如果位置大于 0 则表示包含。本例代码若改为 Instr+Join 组合，那么代码如下。

```
Sub 判断A1到E1是否包含上海()  '放置位置：模块中
    Dim Arr
    Arr = WorksheetFunction.Transpose(WorksheetFunction.Transpose (Range ("A1:E1")))
    MsgBox IIf(InStr("@" & Join(Arr, "@") & "@", "@上海@") > 0, "包含", "不包含")
End Sub
```

可以将以上代码中的“@”理解为匹配方式：在不使用“@”时表示模糊匹配；在使用“@”时，表示精确匹配。

例如当 A1:E1 区域的值分别是北京、上海滩、香港、重庆、成都时，那么使用“@”后代码的执行结果是“不包含”。因为 Join 函数的合并结果是“@北京@上海滩@香港@重庆@成都@”，在其中查找“@上海@”时将返回 0，因为不存在“@上海@”。但是如果不使用“@”，那么 Join 函数的合并结果是“北京上海滩香港重庆成都”，在此字符串中查找“上海”，其结果为 3。

3．本例为了保持代码的通用性，在对变量 Arr1 和变量 Arr2 赋值时采用了 Range. End 属性来定位最后一个单元格，从而使 A、B 列的数据增减时，代码可以自动识别数据区域。

本例案例文件请参考：..\ 第 10 章\ 10-12 获取两列数据的相同项.xlsm

10.3.7 罗列至少三科不及格的学生姓名

【案例要求】图 10-41 为包含若干学生的成绩表，要求在 M 列罗列出所有至少三科不及格的学生姓名。

	A	B	C	D	E	F	G	H	I	J	K	L
1	姓名	语文	数学	地理	化学	英语	政治	体育	计算机	代数	几何	法律
2	计尚云	90	77	93	98	47	72	98	69	71	83	76
3	赵国	55	58	61	88	97	78	84	77	98	89	90
4	罗至贵	100	75	67	46	91	53	58	77	65	66	43
5	徐大鹏	58	80	98	97	77	46	90	43	58	56	98
6	张志坚	41	70	73	86	97	44	50	48	65	51	67
7	朱千文	78	43	47	62	99	42	62	54	83	43	99
8	赵秀文	89	49	64	42	70	80	65	79	95	100	46
9	梁爱国	94	74	43	99	52	100	95	58	97	97	91
10	梁兴	62	53	94	100	55	98	91	93	75	55	60
11	陈随机	68	87	97	82	62	64	64	74	53	87	82

图 10-41　成绩表

【知识要点】ReDim、UBound、Resize 和 For Next。

【程序代码】

```
Sub 罗列至少三科不及格人员姓名()                    '放置位置：模块中
    Dim Arr, Arr2(), RowCount As Integer, ColumnCount As Integer
    Dim 不及格科目 As Byte, 不及格人数 As Integer '声明变量
    Arr = ActiveSheet.UsedRange                     '将活动工作表的已用区域赋值给变量 Arr
  '重置数组 Arr2 的维数和上标，由于是第一次重置，所以第一维的上标允许使用表达式
   ReDim Arr2(1 To UBound(Arr), 1 To 1)
   For RowCount = 2 To UBound(Arr)                  '遍历数组 Arr 的每一行（标题除外）
      '每次进入子循环前将变量的值初始化为 0，此变量代表每个学生的不及格科目
      不及格科目 = 0
      For ColumnCount = 2 To UBound(Arr, 2)          '遍历数组的每一列（标题列除外）
        '如果数组 Arr 的第 RowCount 行第 ColumnCount 列的值小于 60
         If Arr(RowCount, ColumnCount) < 60 Then
         '累加变量的值，此变量表示当前学生的不及格科目数量
         不及格科目 = 不及格科目 + 1
          If 不及格科目 = 3 Then                     '如果不及格科目等于 3,那么
          '累加变量的值，此变量表示不及格科目大于等于 3 的总人数
            不及格人数 = 不及格人数 + 1
          '数组 Arr 第一列的值传递到 Arr2 中（Arr 的第 1 列是姓名）
            Arr2(不及格人数, 1) = Arr(RowCount, 1)
          '退出循环（已经有三科不及格就不需要判断剩下的成绩了）
              Exit For
              End If
         End If
      Next ColumnCount
   Next RowCount
   Cells(1, 13) = "三科不及格人数"                   '在 M1 单元格写入标题
   Cells(2, 13).Resize(不及格人数, 1) = Arr2          '将数组 Arr2 的值导出到 M 列中
End Sub
```

当执行以上过程后，成绩表中不及格科目超过三科的学生姓名将自动罗列在 M 列中，如图 10-42 所示。

	I	J	K	L	M
1	计算机	代数	几何	法律	三科不及格人数
2	69	71	83	76	罗至贵
3	77	98	89	90	徐大鹏
4	77	65	66	43	张志坚
5	43	58	56	98	朱千文
6	48	65	51	67	赵秀文
7	54	83	43	99	梁爱国
8	79	95	100	46	梁兴
9	58	97	97	91	刘子中
10	93	75	55	60	诸有光
11	74	53	87	82	周华章

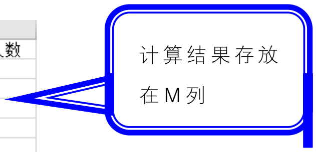


图 10-42　在 M 列罗列出至少三科不及格学生姓名

【代码分析】

1．本例使用了两个数组，一个用于保存成绩表已用区域中的所有信息，包括姓名、科目和成绩，另一个用于保存至少三科成绩不及格者的姓名。所以声明第一个变量 Arr 时采用变体型变量，不需要指定数组维数，对它赋值为已用数据区域的值后它会自动成为数组，其维数和每一维的上标由已用数据区域决定；第二个数组 Arr2 用于保存至少三科不及格的学生姓名，在声明变量时不需要指定数组的维数，而是使用 ReDim 重新指定其维数和上标。

数组 Arr2 的值最终需要存放在 M 列，它是一个纵向的数组，所以在使用 ReDim 语句指定数组的维数时采用了多行一列的二维数组，其行数由数组 Arr 的行数决定。当然，也可以声明为横向的一维数组，赋值完成后通过 Transpose 函数转换方向，然后将数组的值写入到成绩表中。

2．本例中既要横向循环，检查每个学生的所有科目的成绩，又要纵向循环，逐一遍历所有学生，所以使用两个 For Next 循环嵌套。在两层循环中各使用了一个计数器（即符合条件时就累加一次的变量），里层的计数器“不及格科目”用于统计每一名学生小于 60 的成绩个数，当里层循环结束后需要利用它参与新一轮循环。为了避免该变量在上一轮循环中保留下来的值累加到新一轮循环中从而影响判断结果，所以在进入里层的循环之前需要将变量“不及格科目”的值强制恢复为 0。

3．本例在对数组变量 Arr2 指定上标时采取了偷懒的做法，并不是有几名至少三科不及格的学生就将数组的上标定义为几，而是直接将它定义为学生总人数。这种做法在本例中并没有产生不良后果。最后在赋值的区域中使用了“Cells(2, 13).Resize(不及格人数, 1)”，其中变量“不及格人数”刚好等于数组中非空元素的个数，所以在实际赋值时自动略过了数组中的空值，不会对工作表的数据造成破坏。

如果将本例的要求修改为“将数组 Arr2 的值显示在列表框中”，那么为了避免列表框的尾部产生若干空白行，则只能使用 ReDim Preserve 语句根据实际条件指定数组上标。

4．在本例的里层循环语句中，由于只要不及格科目累加到 3 时就已经符合条件，没有必要继续循环下去，此时直接使用“Exit For”语句终止循环即可。由于“Exit For”语句只能作用于当前层的循环，所以不用担心它会影响外层循环，但若使用“Exit Sub”或者“End”语句则必定影响外层循环。

本例案例文件请参考：..\ 第 10 章\ 10-13 罗列至少三科不及格的学生姓名.xlsm

10.4 课后思考

1．在 B1 单元格显示 A1:A10 区域的所有数据，用什么代码实现?

2．判断 A1:A10 区域是否包含“上海”二字，用什么代码实现?

3．假设在工作簿中有 10 个工作表，要同时选中其中的第 1 个、第 5 个和第 8 个工作表，用什么代码实现?

4．计算 A1:A10 与 B2:B20 区域中的相同项，将结果存放在 C 列。

5．使用数组的思路查找活动工作表中已用区域的最小值。

第 11 章　集合与字典的应用

集合 Collection 与字典 Dictionary 都有从重复值中提取唯一值的功能，不过字典对象的提取功能比集合更方便、更快捷，因此在工作中使用字典的概率远比集合大。

对于新人而言，理解集合与字典比较困难，因此有必要详细讲述其语法，并通过多个案例演示其使用思路。

本章要点

- Collection:集合
- Dictionary:字典

11.1　Collection:集合

Collection 对象也被称为集合，它表示多个项目组成的有序集合，可以向集合中随意添加新的元素，也可以随意引用集合中的元素。

更重要的是集合中的所有元素的键值（别名）都具有唯一性，所以可以采用 Collection 对象来获取任意形式数据的唯一值。

11.1.1　集合的特性

可以将集合对象理解为一个包含两组信息的一维数组，一组信息是数组中的值，即集合成员，另一组信息是每个成员的别名（键值），可以通过别名访问该成员的值。其中成员值是允许重复的，但是别名必须具有唯一性，否则无法正确访问集合的值。

例如，如果将区域 A1:H1 看作是一个容器，相当于集合对象，那么 A1:H1 区域中的每个单元格的值就是集合的成员，而每个单元格的地址则是成员的别名。

在区域中，单元格与单元格之间允许存在重复值，但是单元格的地址永远不会重复。

访问 A1:H1 区域的值时既可以使用索引号，也可以使用单元格地址。例如区域中第 5 个值有以下两种表示方式。

Range("A1:H1")(5)——使用索引号 5 引用值，表示取区域中的第 5 个值。

Range("E1")——使用地址引用值，表示取 E1 单元格的值。

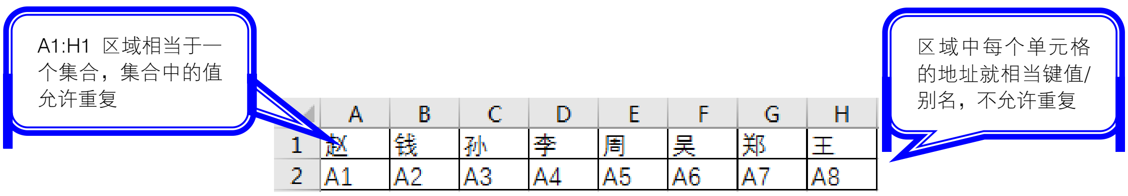


图 11-1　用区域比较集合

集合与上面的 A1:H1 区域一样，在集合中元素的值允许重复，但每个元素的键值不允许重复，

所以才能既通过索引号获取集合中的元素又能以键值来引用集合中的元素。

正是基于键值具有唯一性这个特征，所以常用集合对象来获取一组或者多组数据的唯一值。

11.1.2 集合的语法

集合 Collection 是一个对象，需要创建对象才能使用。此对象有三个方法、一个属性，本节对这些知识点一一详解。

1．创建集合对象

集合 Collection 是一个对象，建立对象有两种方法，一是使用 Dim 语句声明一个 Collection 对象，例如声明一个名为 X 的集合对象，其语法如下。

```
Dim X As New Collection
```

二是使用 With 语法创建，不需要变量，其语法如下。

```
With New Collection

End With
```

当创建对象后，就可以使用 Add 方法向集合中添加成员了。

2．Count 属性

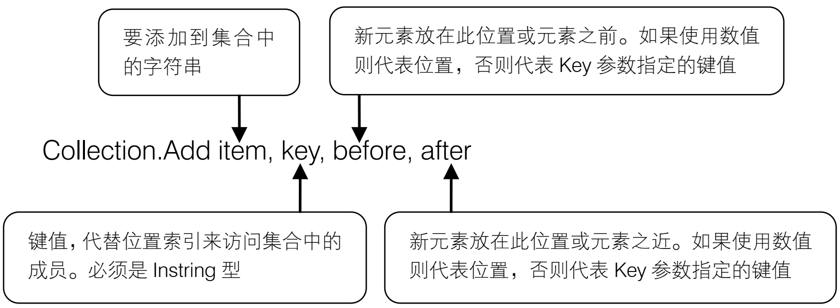
集合的 Count 属性表示集合中的元素个数，其语法如下。

```
object.Count
```

其中 object 代表集合对象。

3．Add 方法

集合的 Add 方法用于向集合中添加一个新的元素，其语法如下。



其中 item 参数是必选参数，其余三个参数是可选参数，通常使用前两个参数即可。

item 参数的值允许重复，key 参数的值不允许重复。可以简单地理解为 item 是值，key 是键值，健值具有唯一性，而值则可以重复。

根据语法说明，向集合中添加赵、钱、孙三个元素，可用以下代码。

```
Sub 向集合中添加三个元素()          '放置位置：模块中
    Dim 集合 As New Collection      '创建一个集合对象
    集合.Add "赵"                   '添加第一个元素
    集合.Add "钱"                   '添加第二个元素
    集合.Add "孙"                   '添加第三个元素，这三个元素是允许重复的
End Sub
```

此时可以使用以下语句测试集合中的元素个数，判断是否添加成功。

```
    MsgBox 集合.Count
```

执行代码后可得到结果 3，表示对象“集合”中包含三个元素。

不过在实际工作中并没有人会以如此方式使用集合，因为仅仅向集合中添加成员并不能发挥集合的优势。在工作中使用集合通常是为了对数据源取唯一值，而不是像普通数组那样仅用于保存一组数据，所以向集合中添加元素时通常需要使用 key 参数，为每个成员指定键值。不过由于键值具有唯一性，在添加重复值时必定出错，所以需要配合“On Error Resume Next”语句使用。

假设 A1:A10 区域的值存在重复性，在利用集合获取该区域中的唯一值时可采用以下代码。

```
Sub 提取 A1 到 A10 区域的唯一值()                     '放置位置：模块中
  Dim 集合 As New Collection                          '创建一个集合对象
  Dim RowCount As Byte                                '声明一个 Byte 型的变量
  On Error Resume Next                                '当代码出错时继续执行下一句代码
  For RowCount = 1 To 10                              '启动循环，从 1 到 10
    集合.Add Cells(RowCount, 1), Cells(RowCount, 1)   '将 A1:A10 区域的值逐个添加到集合中
  Next
End Sub
```

此过程的工作原理如下。

将 A1:A10 区域的值添加到集合中时，同时使用该单元格的值作为键值。由于键值重复时将导致程序出错，所以过程中需要添加“On Error Resume Next”语句，从而让程序得以继续执行，同时又能忽略重复值。

可以使用以下代码获取集合中键值的个数。

```
    MsgBox 集合.Count
```

假设 A1:A10 区域有如图 11-2 所示的数据，那么执行过程可得到集合中唯一值的数量为 7。

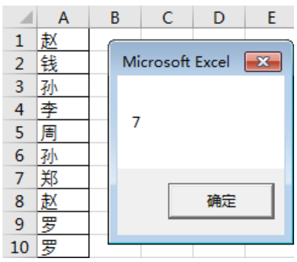


图 11–2　计算区域中的唯一值个数

有两点需要特别指出，其一是 Collection.Add 方法的第二参数 key 必须是文本，所以如果 A1:A10 区域中有数值，那么必须使用 Cstr 函数将它转换成 String 型，否则在代码执行过程中会出错。其二是当区域较大时通过循环语句读取单元格的值效率偏低，需要配合数组的知识应用，最终完善的计算 A1 到 A10 区域的唯一值数量的代码如下。

```
Sub 计算 A1 到 A10 区域的唯一值数量()       '放置位置：模块中
     Dim 集合 As New Collection            '创建一个集合对象
     Dim Arr As Variant                    '声明一个变体型变量，用于保存区域中的值
     Dim RowCount As Byte                  '声明一个 Byte 型的变量，用于 For Next 循环
     On Error Resume Next                  '当代码出错时继续执行下一句代码
     Arr = Range("a1:a10").Value           '将区域的值赋予变量 Arr
     For RowCount = 1 To UBound(Arr)       '启动循环，从 1 到数组第一维的上标
       '将 A1:A10 区域的值逐个添加到集合中
        集合.Add Arr(RowCount, 1), CStr(Arr(RowCount, 1))
     Next
```

```
    MsgBox "唯一值数量:" & 集合.Count      '报告唯一值的数量，即重复出现只算一次
End Sub
```

4．Item 方法

在集合中，Item 方法表示利用位置或键值（别名）返回 Collection 对象的指定成员。

Item 方法的语法如下。

```
object.Item(index)
```

其中 object 表示集合，参数 index 既可以是文本也可以是数值，当使用文本时表示引用键值为 index 的值，当使用数值时表示引用索引号为 index 的值。

```
Sub 通过 Item 方法引用集合成员()               '放置位置：模块中
    Dim 集合 As New Collection                '创建一个集合对象
    集合.Add "宋江", "及时雨"                  '添加第一个元素
    集合.Add "孙二娘", "母夜叉"                '添加第二个元素
    集合.Add "宋江", "呼保义"                  '添加第三个元素（成员值允许重复，键值不允许重复）
    集合.Add "鲁智深", "花和尚"                '添加第四个元素
    集合.Add "林冲", "豹子头"                  '添加第五个元素
    MsgBox 集合.Item(1)                       '通过 Item 方法引用索引号为 1 的成员值
    MsgBox 集合.Item(3)                       '通过 Item 方法引用索引号为 3 的成员值
    MsgBox 集合.Item("及时雨")                '通过 Item 方法引用键值(别名)为“及时雨”的成员值
    MsgBox 集合.Item("呼保义")                '通过 Item 方法引用键值(别名)为“呼保义”的成员值
End Sub
```

以上过程对集合添加了五个元素，其中有两个成员值重复，重复值为“宋江”，不过其键值不同，一个是“及时雨”，一个是“呼保义”。

由于有两个“宋江”，那么可以采用索引号和键值作为区分。

当不知道“宋江”的位置时，如果采用索引号引用集合中的值则需要配合循环语句才行，而采用键值来引用成员值则简单快捷，直接用“集合.Item("及时雨")”或者“集合.Item("呼保义")”即可。

Item 方法允许简写，即可以直接使用索引号，忽略 Item 本身。例如以上过程中的“集合.Item(1)”可以简写成“集合(1)”，其功能完全一致。

5．Remove 方法

Remove 方法用于删除集合中的值，其语法如下。

```
object.Remove index
```

其中参数 index 既可以是文本也可以是数值，当使用文本时表示引用键值为 index 的值，当使用数值时表示引用索引号为 index 的值。

以下过程先用 Add 方法向集合中添加五个元素，再用 Remove 方法删除其中两个值为“宋江”的成员，通过“集合.Count”属性可以验证是否成功删除。

```
Sub 通过 Remove 方法删除集合成员() '放置位置：模块中
  Dim 集合 As New Collection        '创建一个集合对象
  集合.Add "宋江", "及时雨"          '添加第一个元素
  集合.Add "孙二娘", "母夜叉"        '添加第二个元素
  集合.Add "宋江", "呼保义"          '添加第三个元素（成员值允许重复，键值不允许重复）
  集合.Add "鲁智深", "花和尚"        '添加第四个元素
  集合.Add "林冲", "豹子头"          '添加第五个元素
  MsgBox 集合.Count                 '计算集合中的成员数量（结果为 5）
  集合.Remove 1                     '删除第一个成员值（即第一个宋江）
```

```
    集合.Remove "呼保义"                '删除名为“呼保义”的成员值
    MsgBox 集合.Count                   '再次计算集合中的成员数量（结果为 3）
End Sub
```

本例案例文件请参考：..\ 第 11 章\ 11-1 集合 Collection 的语法.xlsm

11.1.3　使用集合获取区域中的不重复值

【案例要求】获取选区中的唯一值，保存在用户指定的区域中。

【知识要点】集合 Collection、数组和 For Each…Next 循环。

【程序代码】

```
Sub 获取选区的唯一值()                          '放置位置：模块中
     Dim Rng As Range                           '声明一个 Range 型的变量
    '将当前选区与已用数据区域的交集赋予变量 Rng，将它作为取唯一值的操作对象。取交集是为
'避免用户全选工作表后执行过程导致 Excel 进入假死状态
    Set Rng = Intersect(ActiveWindow.RangeSelection, ActiveSheet.UsedRange)
    If Rng Is Nothing Then End                  '如果不存在交集则结束过程
    Dim 集合 As New Collection                  '创建一个集合对象
    Dim Arr As Variant                          '声明一个变体型变量，用于保存选区中的值
    Arr = Rng.Value                             '将 Rng 的值赋予数组 Arr，将它作为操作对象
    Dim a As Variant                            '声明一个变体型变量，用于 For Each Next 循环语句
    On Error Resume Next                        '当代码出错时继续执行下一句代码
    For Each a In Arr                           '遍历数组中每一个元素
      '将数组中每个元素添加到集合中，且用元素的值作为集合成员的健值
        集合.Add a, CStr(a)
    Next
    Err.Clear                                   '清除错误值，避免影响后面的操作
    '让用户指定唯一值的存放区域
    Set Rng = Application.InputBox("请选择唯一值保存地址，单个单元格即可", "目标区
域", , , , , , 8)
    If Err <> 0 Then End                        '如果有错误（单击“取消”按钮），那么结束过程
  '声明一个 String 型的数组变量，集合中的值导出到此数组中，再将数组的值一次性导入区域
  '如果直接将集合中的值逐个写入到单元格，程序的执行效率更低，所以采用数组变量作为中转站
    Dim Arr1() As String
    ReDim Arr1(1 To 集合.Count)                 '重置数组变量的维数与上标
    Dim i As Long                               '声明一个 Long 型变量，作为循环语句的计数器
    For Each a In 集合                          '遍历集合中的所有元素
        i = i + 1                               '累加计数器
         If Len(a) > 0 Then Arr1(i) = a         '将集合中每个长度大于 0 的成员值赋值给数组
    Next
    '将数组转置方向后导入 Rng 单元格开始的区域
    Rng.Resize(集合.Count, 1) = WorksheetFunction.Transpose(Arr1)
End Sub
```

执行以上过程后会弹出如图 11-3 所示的对话框，此时单击 F1 单元格，在对话框中将自动产生单元格地址，程序会将当前选区的唯一值列表保存在以 F1 单元格开始的单列区域中，结果如图 11-4 所示。

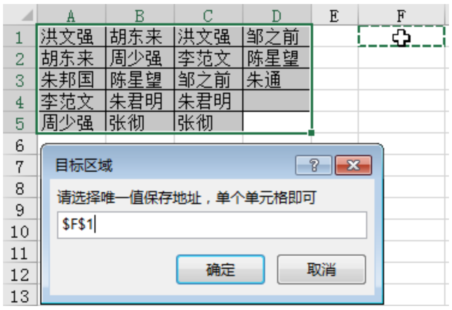


图 11–3　指定唯一值列表存放区域



	A	B	C	D	E	F
1	洪文强	胡东来	洪文强	邹之前		洪文强
2	胡东来	周少强	李范文	陈星望		胡东来
3	朱邦国	陈星望	邹之前	朱通		朱邦国
4	李范文	朱君明	朱君明			李范文
5	周少强	张彻	张彻			周少强
6						陈星望
7						朱君明
8						张彻
9						邹之前
10						朱通

图 11–4　取值结果

【知识补充】

（1）读取区域的值或者向区域中批量写入值，任何时候都有必要使用数组替换区域，从而加快程序执行效率。本例中第一个数组 Arr 用于提高读取速度，第二个数组 Arr1 的功能用于提高写入速度。

（2）获取选区时有两种方式，包括 ActiveWindow.RangeSelection 和 Selection。选择 ActiveWindow.RangeSelection 更为保险，因为在用户选择图片后，Selection 对象代表图片，而 ActiveWindow.RangeSelection 仍然代表区域。

（3）为了加快程序的执行速度，操作对象宜用 ActiveWindow.RangeSelection 与 ActiveSheet.UsedRange 的交集，以避免在用户全选工作表状态下执行过程导致程序进入假死状态。

（4）向集合中写入值或者读取值都只能逐个操作，不如数组方便，所以本例中以数组变量作为中转站，将集合中的值导出为数组变量后，再一次性将数组中的值导入工作表。

本例案例文件请参考：..\ 第 11 章\ 11-2 使用集合获取区域中的不重复值.xlsm

11.1.4　罗列 B 列重复出现的身份证号码

【案例要求】将 B 列重复出现的身份证号码与地址罗列在 C、D 列中。

【知识要点】集合 Collection 和数组与 For Next 循环。

【程序代码】

```
Sub 将B列的重复身份证号码罗列在C列值2()          '放置位置：模块中
    Dim Rng As Range                              '声明一个 Range 型的变量
  '将当前选区与已用数据区域的交集赋予变量 Rng，将它作为取唯一值的操作对象
  '取交集是为避免用户全选工作表后执行过程导致 Excel 进入假死状态
   Set Rng = Intersect(Range("B:B"), ActiveSheet.UsedRange)
   If Rng Is Nothing Then End                     '如果不存在交集则结束过程
   Dim Arr As Variant                             '声明一个变体型变量，用于保存选区中的值
   Arr = Rng.Value                                '将 Rng 的值赋予变量 Arr，将它作为操作对象
   Dim 集合 As New Collection                     '创建一个集合对象
   Dim Arr1() As String                           '声明一个数组变量，存放重复的身份证号码和地址
   Dim RowNum As Integer                          '声明一个 Integer 变量，作为 For Next 循环的变量
   Dim iCount As Integer                          '声明一个 Integer 变量，用于统计重复身份证个数
   On Error Resume Next                           '当代码出错时继续执行下一句代码
   For RowNum = 1 To UBound(Arr)                  '遍历数组中每一个元素
     '将数组中每个元素添加到集合中，且用元素的值作为集合成员的别名
       集合.Add Arr(RowNum, 1), CStr(Arr(RowNum, 1))
       If Err <> 0 Then                           '如果有错误（重复时才出错）
```

```
                iCount = iCount + 1                    '累加计数器，此变量的值等于重复身份证的数量
                '重新指定数组变量 Arr1 的维数和上标，最末一维的上标等于重复身份证个数
                ReDim Preserve Arr1(1 To 2, 1 To iCount)
               '将重复身份证号写入到数组 Arr1 的第 1 行中（前面必须加半角的单引号）
                Arr1(1, iCount) = "'" & Arr(RowNum, 1)
                '将重复身份证的单元格地址转入数组的第 2 行中
                Arr1(2, iCount) = Cells(RowNum , 2).Address
                Err.Clear                              '清除错误，避免影响下一次判断
            End If
        Next
        Range("C1:D1") = Array("身份证号码", "地址") '写入标题
        '将数组变量 Arr1 中的值一次性导入工作表
        Range("C2").Resize(iCount, 2) = WorksheetFunction.Transpose(Arr1)
    End Sub
```

假设在工作表中有如图 11-5 中 A 列和 B 列所示的数据，那么执行过程后将产生如 C 列和 D 列所示的结果。

	A	B	C	D
1	姓名	身份证	身份证号码	地址
2	计尚云	511025197703166171	445122196708032735	$B$5
3	赵国	445122196708032735	422202198512295720	$B$10
4	罗至贵	360429198306262752		
5	徐大鹏	445122196708032735		
6	张志坚	510129196902104913		
7	朱千文	513101197007015051		
8	赵秀文	422202198512295720		
9	梁爱国	340702197701152017		
10	梁兴	422202198512295720		
11	陈随机	422202198808055718		

图 11–5　重复身份证号码提取结果（身份证号码为虚构）

【知识补充】

1．身份证号码在 B 列，所以采用 Intersect(Range("B:B"), ActiveSheet.UsedRange)作为操作对象，也可以采用 Range.End 属性来定位最后一个身份证号码所在单元格。

2．本例是取重复出现的值，而不是唯一值，所以不再用集合对象中的值作为最终结果，而是通过判断在添加身份证号码到集合中去这个过程中是否出错来决定取哪一个值。如果过程执行时出错表示集合中已经存在此身份证号码，所以将它转入到中转站 Arr1 的第一行中，将身份证号码所在单元格的地址写入到第 2 行中。待循环完成后再将中转站 Arr1 的值导出到工作表中。

3．由于“ReDim Preserve”语句只能重置数组变量的最末维的上标，所以在声明变量时采用横向的两行多列的二维数组，循环完成后再通过转置函数 Transpose 转换方向，然后导出工作表。

4．身份证号码长度大于 15 位，而 Excel 在处理数值时只能正确处理 15 位，所以将身份证号码导入单元格之前需要在身份证号码前添加半角的单引号“'”，从而将它转换成文本，确保身份证号码能正确显示。

本例案例文件请参考：..\ 第 11 章\ 11-3 罗列 B 列重复出现的身份证号码.xlsm

使用集合获取唯一值比高级筛选、删除重复项和数据透视表灵活得多，速度上也有较大的优势，而且没有行列限制，没有区域限制，可对任意区域的值取唯一值。

不过集合也有两个明显的缺点：

其一，在集合中生成的唯一值列表只能逐个提取出来，而不像数组那样可以批量导出。

其二，一旦将值写入到集合中就不能修改。例如集合中第三个值是“宋江”，需要将它修改为“及时雨”，没有集合对象的修改方法，只能先删除该成员的值，然后在第二个成员之后插入一个新值，操作上不够便利。

字典也可以实现获取任意区域的唯一值列表，而且可以一次性导入工作表，能够随意修改其中任意成员的值，所以它可以弥补集合对象的缺点。

11.2 Dictionary:字典

Dictionary，也就是字典，它是包含两个一维数组的对象，其中第一行为关键字，第二行是条目，关键字与条目总是一一对应的，形成一个条目对。其中关键字具有唯一性，所以通常采用字典对象替代集合对象来获取区域中的唯一值。

11.2.1 字典对象的前期绑定和后期绑定

字典对象属多种脚本语言中的一种，它集成在动态链接库文件 Scrrun.dll 中，非 Excel 自带功能，必须先注册文件然后调用文件中的资源。

1．注册文件

在默认状态下，Windows 已经注册 Scrrun.dll 文件，不过也不排除误删文件的可能性，所以可以通过手动注册的方式来判断文件是否存在。操作方法如下：

（1）使用【Win+r】组合键打开“运行”对话框；

（2）输入以下命令，然后单击“确定”按钮。

```
Regsvr32 Scrrun.Dll
```

如果弹出如图 11-6 所示的对话框，那么表示文件存在，已注册成功。

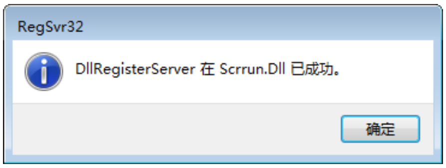


图 11–6　注册 Scrrun.dll

2．前期绑定

注册文件后，还需要绑定文件才可以使用 Scrrun.dll 文件中的资源。有“前期绑定”和“后期绑定”两种方式。

“前期绑定”的操作步骤如下。

（1）选择菜单“工具”→“引用”，打开“引用”对话框；

（2）单击“浏览”按钮，进入系统文件夹（默认路径为“C:\Windows\System32”）找到 Scrrun.dll 文件，然后双击文件返回“引用”对话框；

（3）在“引用”对话框中勾选“Microsoft Scripting Runtim”，结果分别如图 11-7 和图 11-8 所示。

由于32位和64位系统中显示的内容稍有不同，所以此处将两个系统的截图都一并罗列出来。

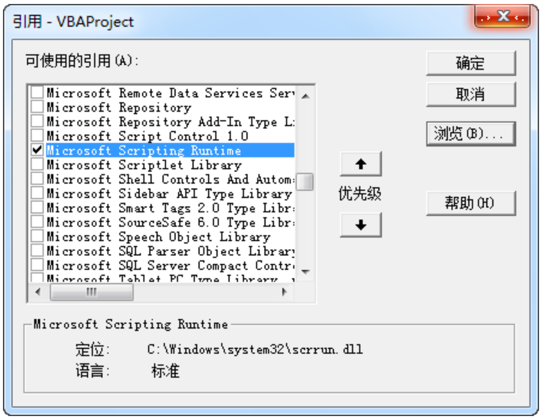


图 11–7　32 位系统中用 Scrrun.dll 文件

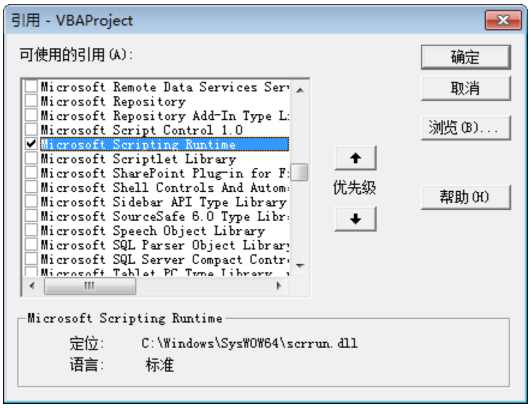


图 11–8　64 位系统中用 Scrrun.dll 文件

在通过“前期绑定”方式引用 Scrrun.dll 文件后，使用 Dim 语句声明字典对象以及编写字典相关的代码时都会自动显示成员列表。

例如编写以下声明字典对象变量的代码时可弹出图 11-9 所示的提示信息。

```
Dim X As New Dictionary
```

继续输入变量名称 x 加小圆点可弹出与字典相关的一切方法和属性，效果如图 11-10 所示。

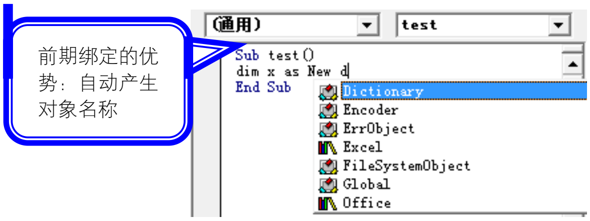


图 11–9　声明变量时的即时提示信息



图 11–10　字典对象的成员列表

代码中的“New Dictionary”表示字典对象，类似于集合对象“New Collection”。

其实，也可以通过代码来引用 Scrrun.dll 文件，操作步骤如下。

（1）在工作表界面选择菜单“开发工具”→“宏安全性”，将弹出“信任中心”的“宏设置”选项卡；

（2）勾选“信任对 VBA 工程对象模型的访问”，然后关闭“信任中心”选项卡；

（3）使用【Alt+F11】快捷键打开 VBE 界面，选择菜单“插入”→“模块”，然后录入以下代码。

```
Sub 自动引用字典对象()          '放置位置：模块中
    On Error Resume Next        '避免不存在 scrrun.dll 文件时出错
   '使用 AddFromFile 方法添加引用
    Application.VBE.ActiveVBProject.References.AddFromFile Environ$("SystemRoot") & "\system32\scrrun.dll"
End Sub
```

执行以上代码后将得到如图 11-7 或图 11-8 所示的同等结果。

3．后期绑定

在“前期绑定”的前提下，编写代码能产生对象的属性与方法列表，有利于程序开发者，却对代码的使用者不利，因为使用代码前必须手动添加引用，否则执行代码时将提示“用户定义类型未定义”错误。

后期绑定的优势在于不需要手动添加引用，可以直接运行代码。

后期绑定字典对象可采用以下代码。

```
CreateObject("scripting.dictionary")
```

在编程时，以上创建字典对象的代码有两种应用方式，包括使用变量和使用 With 语句。

如果通过变量 dic 调用字典对象，那么可采用以下代码模板。

```
Sub test()
    Dim dic As Object
    Set dic = CreateObject("scripting.dictionary")
   '...更多代码...
End Sub
```

如果使用 With 语句，则可采用以下代码模板。

```
Sub test()
    With CreateObject("scripting.dictionary")
        '…更多代码…
    End With
End Sub
```

使用后期绑定字典对象时，书写代码没有对象的成员提示，即不产生属性与方法列表，每个字母必须手动输入。不过编程并不需要每句代码都手动输入，常用的代码可以做成模板备份，放在笔记中，当后续需要使用时从笔记中复制代码即可，相同部分直接使用，不同部分稍加修改，从而可以减少重复的工作。

11.2.2　字典的特点

字典对象用于存放条目对，可以理解为字典中包含两个一一对应的一维数组，但不能理解为二维数组。

第一个数组中是 Key，即关键字，不允许重复；第二个数组 Item，即条目，也称之为项，允许重复。关键字与条目总是一一对应，图 11-11 可以比较形象地展示字典这个容器储存数据的方式。

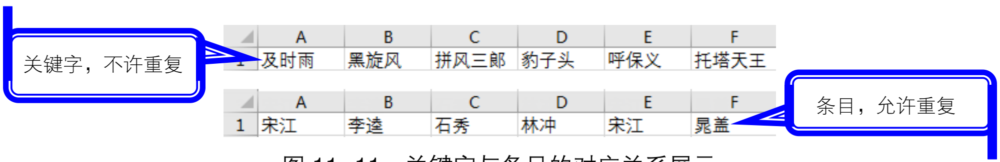


图 11–11　关键字与条目的对应关系展示

字典中的关键字和条目都可以作为一维数组导出来，这是它相较于集合对象的优势。

11.2.3　字典的属性与方法

Dictionary 对象有六个方法和四个属性。其中方法包括：Add、Items、Keys、Remove、RemoveAll 和 Exists，属性则包括：Item、Key、Count 和 CompareMode。

1．Add 方法

功能：添加一对相对应的关键字和条目到 Dictionary 对象，其语法如下。

```
object.Add Key, Item
```

其中 Key 代表关键字，同一个 Dictionary 对象中不允许出现重复的关键字。关键字宜采用文本形式，所以通常书写代码时通过 Cstr 函数将参数转换成 String 型，类似于集合对象 Collection 中 Add 方法的第二参数；Item 参数代表条目，条目允许重复，数据类型不限。

Key 和 Item 两个参数都是必选参数，所以 Add 方法向 Dictionary 对象添加的关键字和条目总是成对的，可通过 Item 方法访问该条目对。

以下过程表示向 Dictionary 对象中添加一对条目，其中关键字是“及时雨”，条目是“宋江”。

```
Sub test()                                      '放置位置：模块中
    With CreateObject("scripting.dictionary")   '创建字典对象
     .Add "及时雨", "宋江"                      '添加一个条目对，关键字是"及时雨"，条目是"宋江"
   End With
End Sub
```

当然有时只需要关键字，不需要条目，那么可以采用以下两种方法处理。

Object.Add "B", Nothing——向条目中添加空对象。

Object.Add "B", ""——向条目中添加空文本。

由于 Key 关键字具有唯一性，所以不能多次添加相同值，否则会出现如图 11-12 所示的提示框。

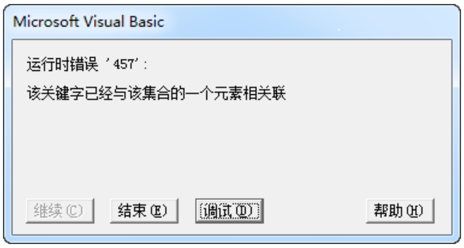


图 11–12　错误提示

本例案例文件请参考：..\ 第 11 章\ 11-4 字典的方法和属性.xlsm

2．Key 属性与 Item 属性

Key 属性的功能是设置字典对象的关键字，可写不可读，即只能通过此属性修改关键字，但不能通过此属性读取关键字，其语法如下。

```
.object.Key(key) = newkey
```

Item 属性的功能是返回或者设置字典对象的条目，可读亦可写，其语法如下。

```
object.Item(key) [= newitem]
```

其中“[= newitem]”部分是可选的，如果忽略该部分则表示读取关键字对应的条目。

在集合对象中添加的成员是不可以修改的，而通过字典对象的 Key 属性与 Item 属性可修改关键字与条目，所以字典较集合有着更多的优越性。

以下过程中包括使用 Add 方法创建条目对、使用 Item 属性修改与获取关键字所对应的条目、利用 Key 属性修改关键字。可以在选择过程后单击【F8】键逐步运行代码，观察每句代码的执行结果，从而理解 Item 属性与 Key 属性的关系与区别。

```
Sub Item 属性与 Key 属性()                          '放置位置：模块中
     With CreateObject("scripting.dictionary")      '创建字典对象
       '添加一个条目对，关键字是"及时雨"，条目是"宋江"
        .Add "及时雨", "宋江"
        '使用 Item 属性获取关键字所对应的条目，验证刚才添加的条目对
        MsgBox "关键字：及时雨" & Chr(10) & "条      目：" & .Item("及时雨"), vbOKOnly,
".Add ""及时雨"", ""宋江"""
        '使用 Item 属性修改关键字所对应的条目
```

```
                .Item("及时雨") = "呼保义"
                '使用 Item 属性获取关键字所对应的条目，验证刚才修改后的条目对
                MsgBox "关键字：及时雨" & Chr(10) & "条    目：" & .Item("及时雨"), vbOKOnly,
" .Item(""及时雨"") = ""呼保义"""
                '利用 Key 属性修改关键字，Key 属性只有设置作用，没有读取作用
'即不能通过 Key 获取关键字的名称
                .Key("及时雨") = "宋公明"
'使用 Item 属性获取关键字所对应的条目，验证刚才修改后的条目对
                MsgBox "关键字：宋公明" & Chr(10) & "条    目：" & .Item("宋公明"), vbOKOnly,
".Key(""及时雨"") = ""宋公明"""
          End With
End Sub
```

执行过程后将分别弹出三个提示框，每个提示框的标题是代码，提示框的内容是该代码对应的执行结果，提示框分别如图 11-13、图 11-14 和图 11-15 所示。

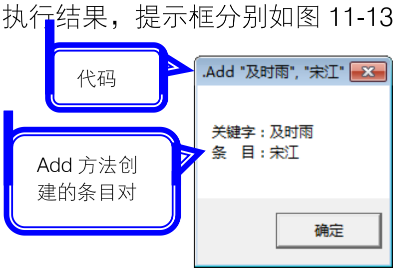


图 11–13　创建条目对的结果

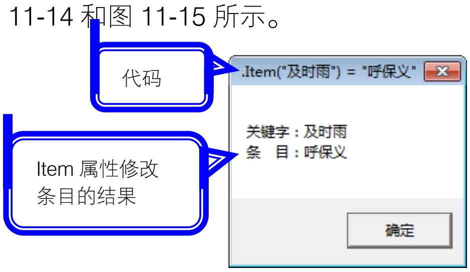


图 11–14　修改条目的结果

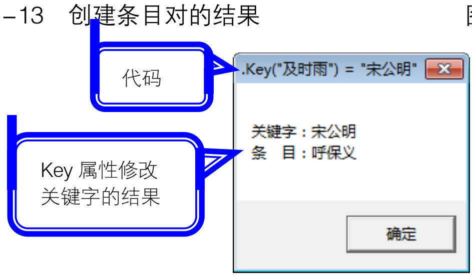


图 11–15　修改关键字的结果

假设在通过 Item 属性修改关键字对应的条目时没有发现参数所指定的关键字，那么代码将会创建一个新的条目对。

例如通过 Add 方法创建了一个关键字为“及时雨”、条目是“宋江”的条目对，然后再通过 Item 属性修改关键字“呼保义”对应的条目，由于字典对象中本不存在名为“呼保义”的关键字，那么它会创建一个新的条目对，代码如下。

```
Sub Item 方法创建条目对()                          '放置位置：模块中
     With CreateObject("scripting.dictionary")  '创建字典对象
       .Add "及时雨", "宋江"                       '添加一个条目对，关键字是“及时雨”"，条目是“宋江”
       .Item("呼保义") = "宋江"                    '添加一个条目对，关键字是“呼保义”，条目是“宋江”
       '分别验证关键字“及时雨”和“呼保义”及其对应的条目
        MsgBox "关键字：及时雨" & Chr(10) & "条    目：" & .Item("及时雨")
        MsgBox "关键字：呼保义" & Chr(10) & "条    目：" & .Item("呼保义")
    End With
End Sub
```

执行以上代码后，字典对象中将存在两个条目对，分别为关键字“及时雨”对应条目“宋江”，关键字“呼保义”对应条目“宋江”。

基于此特性，也可以采用 Item 属性创建字典的条目对，不再局限于 Add 方法，而且还可以将 Item 属性配合循环语句使用，反复修改（累加或者递减）关键字对应的条目的值。

```
Sub Item 方法创建条目对和累加条目()                    '放置位置：模块中
    With CreateObject("scripting.dictionary")          '创建字典对象
        .Item("一车间") = 80                           '添加一个条目对，关键字是"一车间"，条目是 80
        .Item("二车间") = 90                           '添加一个条目对，关键字是"二车间"，条目是 90
        .Item("一车间") = .Item("一车间") + 100        '累加关键字"一车间"的值
        .Item("二车间") = .Item("二车间") + 120        '累加关键字"二车间"的值
        '分别验证关键字"一车间"和"二车间"及其对应的条目
        MsgBox "关键字：一车间" & Chr(10) & "条    目：" & .Item("一车间")
        MsgBox "关键字：二车间" & Chr(10) & "条    目：" & .Item("二车间")
    End With
End Sub
```

以上过程通过 Item 属性创建了两个条目对，然后又通过 Item 属性累加条目的值，最后的结果是关键字“一车间”对应的条目为 180，关键字“二车间”对应的条目为 210。使用此思路可以实现对工作表数据分类汇总，具体操作将在后面的案例中详细展示。

3．Keys 方法与 Items 方法

Keys 方法可返回一个没有重复值的一组数据，该数组包含一个字典对象中的全部关键字，即从字典中获取第一行数据，其语法如下。

```
object.Keys
```

Keys 方法的返回值可以导入区域，也可以赋值给列表框，还可以直接赋值给数组变量，或者通过 Join 函数将它合并为一个字符串。所以在取唯一值方面，字典对象比高级筛选、删除重复项、数据透视表和集合对象都更强大。

Items 方法也能返回一个一维数组，该数组包含一个字典对象中的全部条目，即从字典中获取第二行数据，其语法如下。

```
object.Items
```

以下过程先用 Add 方法向字典中创建四个条目对，然后分别通过 Keys 方法和 Items 方法将两个一维数组输入到 A、B 列。

```
Sub Keys 与 Items 方法输入数组()                       '放置位置：模块中
    With CreateObject("scripting.dictionary")          '创建字典对象
        .Add "及时雨", "宋江"                          '添加第一个条目对
        .Add "黑旋风", "李逵"                          '添加第二个条目对
        .Add "拼命三郎", "石秀"                        '添加第三个条目对
        .Add "呼保义", "宋江"                          '添加第四个条目对
       '将 4 个关键字输入到 A1:A4 区域中
        Range("A1:A4") = WorksheetFunction.Transpose(.Keys)
        '将 4 个条目输入到 B1:B4 区域中
        Range("B1:B4") = WorksheetFunction.Transpose(.Items)
    End With
End Sub
```

由于 Keys 和 Items 的返回值都是一维数组，横向排列，所以导出到 A1:A4 和 B1:B4 区域前需要利用工作表函数 Transpose 转置方向，输出结果如图 11-16 所示。

事实上也可以通过 Join 函数将关键字与条目显示在提示框中，将过程中最后两句修改为以下语句即可，输入结果如图 11-17 所示。

```
MsgBox "关键字：" & Join(.Keys, "，") & Chr(10) & "条    目：" & Join(.Items, "，")
```

	A	B
1	及时雨	宋江
2	黑旋风	李逵
3	拼命三郎	石秀
4	呼保义	宋江

图 11-16　将关键字与条目输出到工作表中

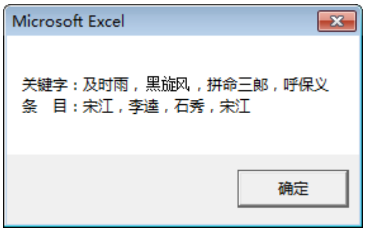


图 11-17　输出关键字与条目到提示框中

Items 和 Keys 都是字典对象的方法而不是属性，它们不支持通过索引号引用单个元素。如果需要获取单个元素则可以借助数组变量实现。

例如获取索引号为 2 的关键字和条目，可在过程中插入以下代码。

```
Dim Arr1, Arr2
Arr1 = .Keys
Arr2 = .Items
MsgBox Arr1(2) & Chr(10) & Arr2(2)
```

4．Remove 方法和 RemoveAll 方法

Remove 方法功能用于从一个字典对象中删除一个关键字和条目，其语法如下。

```
object.Remove(key)
```

Remove 方法一次只能删除一个关键字和条目，而 RemoveAll 方法可以删除所有条目对，其语法如下。

```
object.RemoveAll
```

5．Exists 方法

Exists 方法用于判断 Dictionary 对象中是否包含某个关键字，如果返回值为 True 表示该关键字存在，否则不存在，其语法如下。

```
object.Exists(key)
```

6．Count 属性

Count 属性可返回 Dictionary 对象中的条目数量，其语法如下。

```
object.Count
```

通常在导出字典对象中的关键字或者条目之前需要计算条目数量，例如根据此数量重置区域引用的高度或者宽度，然后再将关键字或者条目导入区域。

前面的 Sub 过程“Keys 与 Items 方法输入数组”中的代码“Range("A1:A4") = WorksheetFunction.Transpose(.Keys)”就有必要修改区域地址 A1:A4，从而提升代码的通用性。修改的思路是先计算关键字的数量，然后根据此数量重置 Range("A1")的高度，从而产生一个新的区域，此区域的单元格个数对应字典中关键字的个数，完整代码如下。

```
Range("A1").Resize(.count,1) = WorksheetFunction.Transpose(.Keys)
```

7．CompareMode 属性

CompareMode 属性用于设置或返回某个字典对象中的比较字符串关键字的比较模式。简单而言是指在比较关键字中的英文字母时，是否需要区分大小写，默认区分大小写。

修改或者返回 CompareMode 属性的语法如下。

```
object.CompareMode[ = compare]
```

在对参数 compare 赋值为 vbBinaryCompare 时表示区分大小写；在赋值为 vbTextCompare

时表示不区分大小写。

11.2.4　获取选区中的唯一值

【案例要求】获取选区中的唯一值，保存在用户指定的区域中。

【知识要点】CreateObject("scripting.dictionary")、Add 方法、Count 属性和 Keys 方法。

【程序代码】

```
Sub 获取选区的唯一值()                          '放置位置：模块中
     Dim Rng As Range                           '声明一个 Range 型的变量
   '将当前选区与已用数据区域的交集赋予变量 Rng，将它作为取唯一值的操作对象
   '取交集是为了避免用户全选工作表后执行过程导致 Excel 进入假死状态
   Set Rng = Intersect(ActiveWindow.RangeSelection, ActiveSheet.UsedRange)
   If Rng Is Nothing Then End                   '如果不存在交集则结束过程
   Dim Arr As Variant                           '声明一个变体型变量，用于保存选区中的值
   Arr = Rng.Value                              '将 Rng 的值赋予变量 Arr，将它作为操作对象
   Dim a As Variant                             '声明一个变体型变量，用于 For Each Next 循环语句
   On Error Resume Next                         '当代码出错时继续执行下一句代码
   With CreateObject("scripting.dictionary")    '创建字典对象
      For Each a In Arr                         '遍历数组中每一个元素
    '将数组中每个长度大于 0 的元素添加到字典的关键字中，其对应的条目为空文本
     If Len(a) > 0 Then .Add CStr(a), ""
   Next a
   Err.Clear                                    '清除错误值，避免影响后面的操作
       '让用户指定结果存放区域
   Set Rng = Application.InputBox("请指定结果放存区域，单个单元格即可", "存放区域", , , , , , 8)
   If Err <> 0 Then End                         '如果有错误（单击了“取消”按钮）则结束过程
       '将关键字导入用户指定的单元格开始的区域，区域的高度由字典的条目对个数决定
     Rng(1).Resize(.Count, 1) = WorksheetFunction.Transpose(.Keys)
   End With
End Sub
```

选择区域后再执行以上过程将弹出如图 11-18 所示的输入框，在其中指定存放区域并单击“确定”按钮后可得到如图 11-19 所示结果。

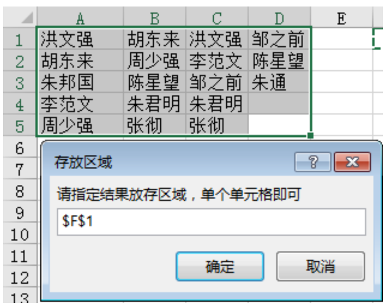


图 11–18　指定结果存放区域



图 11–19　代码执行结果

由于在代码中使用了 Intersect 方法，所以即使选区有很多空白区域也不会浪费资源，程序会自动略过空白区域。

【知识补充】

（1）在引用选区时不宜用 Selection，宜使用 Intersect(ActiveWindow.RangeSelection, ActiveSheet.UsedRange)，其更通用。

（2）本例仅取区域中的唯一值，所以在向字典对象写入值时只需要关键字即可，条目可以采用空文本。

（3）由于在添加关键字时会造成重复，所以必须配合“On Error Resume Next”语句，从而过滤重复值。

（4）Keys 方法产生的值是横向排列的一维数组，而实际储存区域是纵向的，所以需要使用 Transpose 函数转置方向。

本例案例文件请参考：..\ 第 11 章\ 11-5 获取选区中的唯一值.xlsm

11.2.5 对采购表分类求和

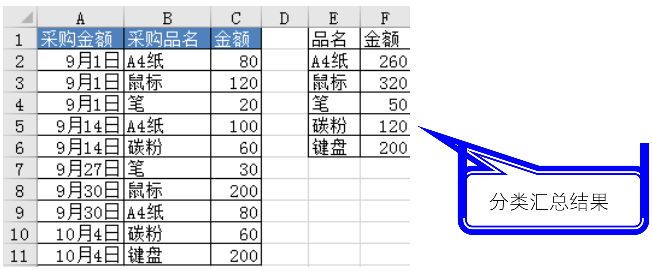
【案例要求】将图 11-20 中 A1:C11 区域的采购表按品名分类汇总，实现 E1:F6 的汇总结果，类似于数据透视表的数据汇总。

【知识要点】CreateObject("scripting.dictionary")、Add、Item、Keys 和 Items。

【程序代码】

```
Sub 对采购表分类求和()                              '放置位置：模块中
  Dim Arr As Variant                                '声明一个变体型变量，用于保存选区中的值
  '将 B2 到 C 列最后一个非空单元格组成的区域赋值给变量 Arr
  Arr = Range(Range("B2"), Cells(Rows.Count, "C").End(xlUp)).Value
  Dim I As Integer                                  '声明一个 Integer 型变量，用于 For Each Next 循环语句
  With CreateObject("scripting.dictionary")         '创建字典对象
    For I = 1 To UBound(Arr)                        '遍历数组中每一行
    '如果字典对象中没有名为 CStr(Arr(I, 1))的关键字，那么创建此关键字，对应的条目为 Arr(I, 2)
    '如果字典对象中有名为 CStr(Arr(I, 1))的关键字，那么对它的条目的值累加 Arr(I, 2)
    '其中 CStr(Arr(I, 1))代表数组中第一列的值，即品名；Arr(I, 2)代表数组中第二列的值，即金额
        If Len(Arr(I, 1)) > 0 Then .Item(CStr(Arr(I, 1))) = .Item(CStr(Arr(I, 1))) + Arr(I, 2)
    Next
    '在 E1:F1 区域写入标题
    Range("E1:F1") = Array("品名", "金额")
    '分别导出字典对象的关键字和条目
    Range("E2").Resize(.Count, 1) = WorksheetFunction.Transpose(.Keys)
    Range("F2").Resize(.Count, 1) = WorksheetFunction.Transpose(.Items)
    '对结果存放区域添加边框
    Range("E1").CurrentRegion.Borders.LineStyle = xlContinuous
  End With
End Sub
```

执行以上过程后将得到如图 11-20 所示结果。



	A	B	C	D	E	F
1	采购金额	采购品名	金额		品名	金额
2	9月1日	A4纸	80		A4纸	260
3	9月1日	鼠标	120		鼠标	320
4	9月1日	笔	20		笔	50
5	9月14日	A4纸	100		碳粉	120
6	9月14日	碳粉	60		键盘	200
7	9月27日	笔	30			
8	9月30日	鼠标	200			
9	9月30日	A4纸	80			
10	10月4日	碳粉	60			
11	10月4日	键盘	200			

图 11-20　将采购表按品名分类汇总

【知识补充】

（1）在本例中参与计算的区域是 B、C 列中标题以外的数据区域，过程中引用该区域的方法是先定位左上角的 B2，再定位 C 列最后一个非空单元格，两者组合形成的矩形区域即为目标区域。所以本例采用了代码“Range(Range("B2"), Cells(Rows.Count, "C").End(xlUp))”引用该区域，而不是直接用 Range("B2:C11")。

（2）代码“.Item(CStr(Arr(I, 1))) = .Item(CStr(Arr(I, 1))) + Arr(I, 2)”对于初学者而言较难理解，可以阅读本书 11.2.3 节案例。二者的思路完全一致，仅仅数据源不同。

如果换一种写法，则可以使代码更易懂，同时代码也会更长。以下代码采用 Exists 方法先判断指定的关键字是否存在，没有时则添加此关键字，如果有则对其条目的值累加新值。

```
    If Len(Arr(I, 1)) > 0 Then                '如果长度大于 0
      If .Exists(CStr(Arr(I, 1))) Then        '如果字典中存在此关键字
          '那么在关键字原来的值的基础上累加 Arr(I, 2)的值
          .Item(CStr(Arr(I, 1))) = .Item(CStr(Arr(I, 1))) + Arr(I, 2)
        Else                                  '否则
          '添加一对条目，关键字为 Arr(I, 1)，条目为 Arr(I, 2)
         .Add CStr(Arr(I, 1)), Arr(I, 2)
        End If
      End If
```

本例案例文件请参考：..\ 第 11 章\ 11-6 对采购表分类求和.xlsm

11.2.6　对采购表分类计数

【案例要求】将图 11-21 中 A1:C21 区域的采购表按品名分类计数，类似于数据透视表的结果。

【知识要点】CreateObject("scripting.dictionary")、Add、Item、Keys 和 Items。

【程序代码】

本例代码和前一个案例的过程“对采购表分类求和”只差几个字。即把以下代码：

```
    If Len(Arr(I, 1)) > 0 Then .Item(CStr(Arr(I, 1))) = .Item(CStr(Arr(I, 1))) + Arr(I, 2)
```

修改为：

```
    If Len(Arr(I, 1)) > 0 Then .Item(CStr(Arr(I, 1))) = .Item(CStr(Arr(I, 1))) + 1
```

【知识补充】

（1）本例要求将 11.2.5 节的“求和”改为“计数”，所以代码的差异仅在于累加的值，其他思路完全一致。

（2）如果将案例要求修改为既要求和也要计数呢？由于字典只有 2 行，所以需要创建两个字典对象，一个对象的条目存放金额合计，另一个存放购买次数即可。完整代码如下：

```
  Sub 对采购表分类求和与计数()                         '放置位置：模块中
    Dim Arr As Variant                               '声明一个变体型变量，用于保存选区中的值
    '将 B2 到 C 列最后一个非空单元格组成的区域赋值给变量 Arr
    Arr = Range(Range("B2"), Cells(Rows.Count, "C").End(xlUp)).Value
    Dim I As Integer                                 '声明一个 Integer 变量，用于 For Each Next 循环
    Dim Dic1 As Object, Dic2 As Object               '声明两个 Object 的对象变量
    Set Dic1 = CreateObject("scripting.dictionary")  '创建字典对象
    Set Dic2 = CreateObject("scripting.dictionary")  '创建字典对象
    For I = 1 To UBound(Arr)                         '遍历数组中的每一行
      '对字典 Dic1 写入金额合计，对 Dic2 写入购买次数
        If Len(Arr(I, 1)) > 0 Then
```

```
            Dic1.Item(CStr(Arr(I, 1))) = Dic1.Item(CStr(Arr(I, 1))) + Arr(I, 2)
            Dic2.Item(CStr(Arr(I, 1))) = Dic2.Item(CStr(Arr(I, 1))) + 1
        End If
    Next
    '在 E1:F1 区域写入标题
    Range("E1:G1") = Array("品名", "金额求和", "购买次数")
    '分别导出字典对象的关键字和条目
    Range("E2").Resize(Dic1.Count, 1) = WorksheetFunction.Transpose (Dic1. Keys)
    Range("F2").Resize(Dic1.Count, 1) = WorksheetFunction.Transpose(Dic1. items)
    Range("G2").Resize(Dic2.Count, 1) = WorksheetFunction.Transpose(Dic2. items)
    '对结果存放区域添加边框
    Range("E1").CurrentRegion.Borders.LineStyle = xlContinuous
End Sub
```

执行以上过程后结果如图 11-21 所示。



	A	B	C	D	E	F	G
1	采购金额	采购品名	金额		品名	金额求和	购买次数
2	9月1日	A4纸	80		A4纸	260	3
3	9月1日	鼠标	120		鼠标	320	2
4	9月1日	笔	20		笔	50	2
5	9月14日	A4纸	100		碳粉	120	2
6	9月14日	碳粉	60		键盘	200	1
7	9月27日	笔	30				
8	9月30日	鼠标	200				
9	9月30日	A4纸	80				
10	10月4日	碳粉	60				
11	10月4日	键盘	200				

图 11–21　同时对产品的购买金额计数与求和

本例案例文件请参考：..\ 第 11 章\ 11-7 对采购表分类计数.xlsm

11.2.7　对产量表按组别和产品分类统计

【案例要求】对图 11-22 的数据按产品品名，将产量和不良品分类汇总，然后按组别对产量和不良品分类汇总，实现类似于数据透视表的效果。

【知识要点】CreateObject("scripting.dictionary")、Count、Add、Item、Keys 和 Items。

【程序代码】

```
Sub 按产品分类统计产量与不良品()                    '放置位置：模块中
    Dim Arr As Variant                               '声明一个变体型变量，用于保存选区中的值
    '将 B2 到 E 列最后一个非空单元格组成的区域赋值给变量 Arr
    Arr = Range(Range("B2"), Cells(Rows.Count, "E").End(xlUp)).Value
    Dim I As Integer                                 '声明一个 Integer 变量，用于 For Each Next 循环
    Dim Dic1 As Object, Dic2 As Object               '声明两个 Object 的对象变量
    Set Dic1 = CreateObject("scripting.dictionary")  '创建字典对象
    Set Dic2 = CreateObject("scripting.dictionary")  '创建字典对象
    For I = 1 To UBound(Arr)                         '遍历数组中每一行
        '对字典 Dic1 写入产量合计，对 Dic2 写入不良品合计
        If Len(Arr(I, 2)) > 0 Then
            Dic1.Item(CStr(Arr(I, 2))) = Dic1.Item(CStr(Arr(I, 2))) + Arr(I, 3)
            Dic2.Item(CStr(Arr(I, 2))) = Dic2.Item(CStr(Arr(I, 2))) + Arr(I, 4)
        End If
    Next
    '在 G1:I1 区域写入标题
    Range("G1:I1") = Array("品名", "产量", "不良品")
    '分别导出字典对象的关键字和条目
```

```
    Range("G2").Resize(Dic1.Count, 1) = WorksheetFunction.Transpose (Dic1. Keys)
    Range("H2").Resize(Dic1.Count, 1) = WorksheetFunction.Transpose(Dic1. items)
    Range("I2").Resize(Dic2.Count, 1) = WorksheetFunction.Transpose(Dic2. items)
    '对结果存放区域添加边框
    Range("G1").CurrentRegion.Borders.LineStyle = xlContinuous
End Sub
```

```
Sub 按组别分类统计产量与不良品()                    '放置位置：模块中
    Dim Arr As Variant                              '声明一个变体型变量，用于保存选区中的值
    '将 B2 到 C 列最后一个非空单元格组成的区域赋值给变量 Arr
    Arr = Range(Range("B2"), Cells(Rows.Count, "E").End(xlUp)).Value
    Dim I As Integer                                '声明一个 Integer 变量，用于 For Each Next 循环
    Dim Dic1 As Object, Dic2 As Object              '声明两个 Object 的对象变量
    Set Dic1 = CreateObject("scripting.dictionary") '创建字典对象
    Set Dic2 = CreateObject("scripting.dictionary") '创建字典对象
    For I = 1 To UBound(Arr)                        '遍历数组中每一行
        '对字典 Dic1 写入产量合计，对 Dic2 写入不良品合计
        If Len(Arr(I, 1)) > 0 Then
            Dic1.Item(CStr(Arr(I, 1))) = Dic1.Item(CStr(Arr(I, 1))) + Arr(I, 3)
            Dic2.Item(CStr(Arr(I, 1))) = Dic2.Item(CStr(Arr(I, 1))) + Arr(I, 4)
        End If
    Next
    '在 E1:F1 区域写入标题
    Range("G1:I1") = Array("品名", "产量", "不良品")
    '分别导出字典对象的关键字和条目
    Range("G2").Resize(Dic1.Count, 1) = WorksheetFunction.Transpose(Dic1. Keys)
    Range("H2").Resize(Dic1.Count, 1) = WorksheetFunction.Transpose(Dic1. items)
    Range("I2").Resize(Dic2.Count, 1) = WorksheetFunction.Transpose(Dic2. items)
    '对结果存放区域添加边框
    Range("G1").CurrentRegion.Borders.LineStyle = xlContinuous
End Sub
```

分别执行以上两段代码，将得到如图 11-23 和图 11-24 所示的汇总结果。

	A	B	C	D	E
1	姓名	组别	产品	产量	不良品
2	计尚云	A组	螺丝	52	6
3	赵国	C组	钳子	71	8
4	罗至贵	A组	改锥	93	1
5	徐大鹏	C组	螺丝	90	6
6	张志坚	A组	梅花刀	69	5
7	朱千文	B组	钳子	99	3
8	赵秀文	C组	梅花刀	94	7
9	梁爱国	A组	螺丝	52	7
10	梁兴	B组	钳子	98	3
11	陈随机	B组	螺丝	68	3

图 11–22　产量表

	F	G	H	I
1		品名	产量	不良品
2		A组	266	19
3		C组	255	21
4		B组	265	9
5				

图 11–23　按产品分类汇总

	F	G	H	I
1		品名	产量	不良品
2		螺丝	262	22
3		钳子	268	14
4		改锥	93	1
5		梅花刀	163	12
6				

图 11–24　按组别分类汇总

本例案例文件请参考：..\ 第 11 章\ 11-8 对产量表按组别和产品分类统计.xlsm

11.3　课后思考

1．在取唯一值时，字典与集合两个对象有何区别?

2．字典对象的 Item 属性和 Items 方法有何区别? Key 属性与 Keys 方法有何区别?

3．在图 11-25 中 A 列到 E 列用于存放日产量数据，需要手动录入。G 列到 I 列用于存放按品名汇总的汇总表，K 列到 M 列用于存放按组别汇总的汇总表。要求录入日产量数据时自动生成两个汇总表，也就是说工作表中原本是空白的，当录入与产量相关的数据时，代码检测到数据后利用录入的数据生成两个汇总表，每更新一行就同步更新汇总结果。

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	姓名	组别	产品	产量	不良品		品名	产量	不良品		组别	产量	不良品
2	刘洪年	A组	镙丝	52	6		镙丝	52	6		A组	266	19
3	赵国	C组	钳子	71	8		钳子	268	14		C组	255	21
4	罗至贵	A组	改锥	93	1		改锥	93	1		B组	265	9
5	徐大鹏	C组	螺丝	90	6		螺丝	210	16				
6	张志坚	A组	梅花刀	69	5		梅花刀	163	12				
7	朱千文	B组	钳子	99	3								
8	赵秀文	C组	梅花刀	94	7								
9	梁爱国	A组	螺丝	52	7								
10	梁兴	B组	钳子	98	3								
11	陈随机	B组	螺丝	68	3								
12													

图 11–25　录入产量报表时自动生成汇总表

4．A1：A40 区域中有重复出现若干次的姓名，要求要罗列出重复出现次数大于 4 的姓名，以及它的出现次数。

5．将图 11-26 中 A1:B9 区域所示数据整合为如 D1:E4 区域所示的结果。

	A	B	C	D	E
1	省	市		省	市
2	四川	成都		四川	成都、内汇、攀枝花
3	湖北	武汉		湖北	武汉、天门、汉口
4	湖北	天门		湖南	长沙、张家界
5	湖南	长沙			
6	四川	内汇			
7	湖南	张家界			
8	湖北	汉口			
9	四川	攀枝花			

图 11–26　合并省市名称

第 12 章 设计程序窗体

Excel 的每个对话框都是一个窗体，窗体通常承载两个重要使命：在屏幕上输出信息和向工作表录入数据。

窗体是一个由窗体容器和放置在容器中的诸多控件组成的界面，窗体和每类控件都有各自的属性和事件。本章主要着眼于窗体与应用在窗体中的多种控件的属性和事件，并通过多个案例展示窗体实战技巧。

本章要点

- 窗体与控件简介
- 设置属性
- 窗体与控件的事件
- 窗体应用实战

12.1 窗体与控件简介

在 VBA 中用 UserForm 表示窗体对象。

窗体总是配合控件使用的，单一的 UserForm 对象没有实用价值。本节逐一介绍 UserForm 对象和常用控件的功能、外观，以及在 VBA 中如何用代码引用这些控件。

12.1.1 窗体的功能

窗体一般是指 UserForm 对象及放置在其中的控件的集合，而非单一的 UserForm 对象。

在工作中窗体是极其有用的工具，善用窗体能为工作提速、确保录入数据的准确度，同时也是开发专业程序或者各种中小型系统的必备工具。

简单而言，窗体的作用分为两类：输入和输出。

1．输出：在屏幕上显示信息

Msgbox 函数就是一个简单的窗体，它可以在屏幕上显示简短的信息，而工作中遇到更复杂的需求时则不得不借助复杂的 UserForm 窗体来实现。

例如在如图 12-1 所示的工作簿中包含“A 组”、“B 组”和“C 组”3 个工作表，工作表中分别存放 3 个组的职工姓名、机台与产量信息，在工作簿中设计了一个查询窗体用于显示查询结果。图中查询目标是 500，所以窗体中的列表框罗列了所有大于 500 的产量信息，以及每条信息所在工作表的名称。

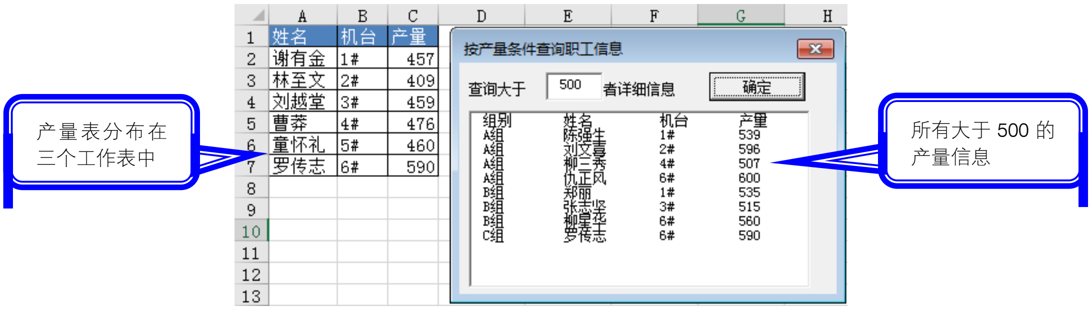


图 12–1　在窗体中显示查询结果

图 12-1 属于比较复杂的信息展示，它可从多个工作表中按条件获取信息，然后将信息通过列表框展示出来。Excel 的内置功能无法满足这种工作需求，所以有必要通过设计窗体来实现，这是窗体的优势之一。读者可以从随书案例中获取这个工作簿，其中包含设计好的窗体以及查询功能的源代码。

本例案例文件请参考：..\ 第 12 章\ 12-1 在窗体中显示跨表查询结果.xlsm

2．输入：将窗体中的信息导入到工作表中

单元格可以通过有效性对录入的值进行校验，但是在有效性方面存在诸多缺点，例如容易被破坏（在粘贴数据时会删除有效性设置）、无法每录入一个字符就校验一次等。如果在窗体中录入数据，那么可以解决关于有效性的所有问题。

另外，当待录入的数据分布在多个工作表中时，需要来回切换工作表，即要使用鼠标又要使用键盘，从而降低录入效率。使用窗体可以解决此问题。

图 12-2 中的工作簿包含 4 个工作表，其中第 4 个工作表储存了职工的基本信息。图 12-3 是资料录入界面，将窗体与“职员信息表”配合应用，可以自动识别工作表名以及录入机台号。在原本录入姓名和产量时需要在多个工作表之间来回切换，而且需要录入每个职员对应的机台号，在使用窗体后可以自动完成这两项工作，从而减少约一半的工作量。



图 12–2　职员信息表

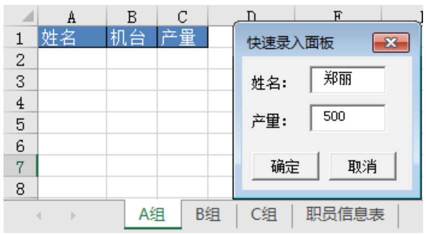


图 12–3　录入资料

当录入错别字时系统还会弹出提示框，如图 12-4 所示。

如果在“产量”文本框中录入了文本时也会弹出提示框，如图 12-5 所示。

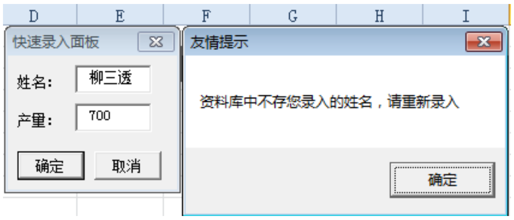


图 12–4　录入错别字时自动提示

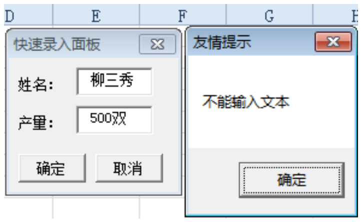


图 12–5　对产量录入文本时自动提示

每录入一笔资料后，窗体将资料导入对应的区域，然后自动清空窗体中的上一笔资料，并定位到下一次需要录入值的文本框中，如图 12-6 所示。这种技术的应用可以避免在键盘录入过程中频繁操作鼠标，从而提高录入数据的速度。

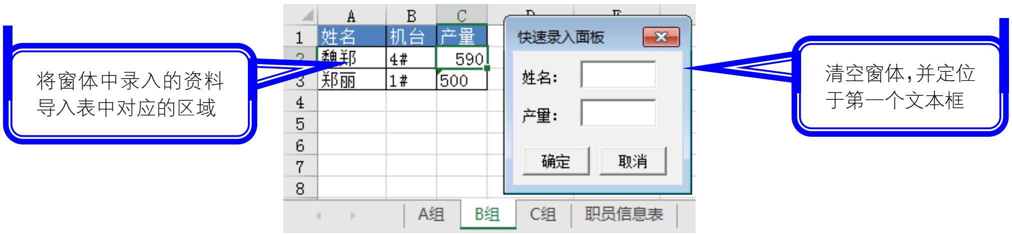


图 12–6　将录入的资料输出到对应的工作表

本例案例文件请参考：..\ 第 12 章\ 12-2 借助窗体跨表录入产量信息.xlsm

12.1.2　创建与运行 UserForm 对象

UserForm 是窗体的主容器，其他一切控件都放置在此容器之中，所以设计窗体的第一个步骤总是插入一个 UserForm 对象。

插入 UserForm 对象的基本步骤如下。

（1）在工作表界面按【Alt+F11】组合键打开 VBE 窗口；

（2）选择菜单“插入”→“用户窗体”，在工程资源管理器的右方将会产生默认名称为“UserForm1”的用户窗体，如图 12-7 所示。

UserForm 对象有“名称”、“BackColor”、“Caption”等属性，在属性窗口中可以看到 UserForm 对象的这些属性，如图 12-8 所示。

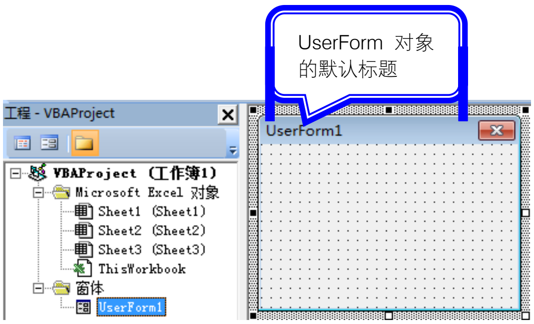


图 12–7　UserForm1 默认界面

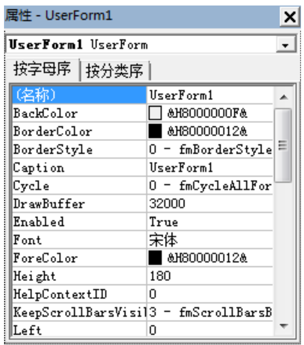


图 12–8　UserForm1 的属性

在实际工作中，插入 UserForm 对象后都要根据需求修改 UserForm 对象的某些窗体属性，

包括左上角显示的文本（Caption 属性）、高度（Height 属性）、宽度（Width 属性）、背景色（BackColor 属性）等。

在插入 UserForm 对象后，若要运行此 UserForm 对象，可通过以下两种方式实现。

1．用快捷键运行窗体

在 VBE 中运行窗体最快捷的方法是在选择窗体后按下【F5】键，窗体将显示在工作表上层，效果如图 12-9 所示。

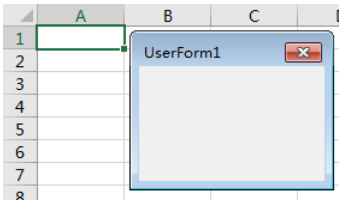


图 12–9　运行窗体

2．用代码调用窗体

用代码显示窗体可调用 Show 方法，操作步骤如下。

（1）选择菜单“插入”→“模块”。

（2）在模块中录入如下代码。

```
Sub 运行窗体()          '放置位置：模块中
 UserForm1.Show         '运行窗体
End Sub
```

（3）执行以上过程，窗体 UserForm1 将显示在工作表上层，位于工作表界面的中心位置。

新创建的 UserForm 对象，其“名称”属性与“Caption”属性往往一致。例如本例所插入的窗体的“名称”和“Caption”属性都是“UserForm1”。通过代码运行窗体时，调用的是 UserForm 对象的“名称”属性，“Caption”属性仅决定 UserForm 对象左上角所显示的标题文字，“Caption”属性值不能代表 UserForm 对象。

假设将窗体的“名称”属性修改为“AA”，将窗体的“Caption”属性修改为“BB”，那么通过代码调用窗体时应使用“AA.Show”而非“BB.Show”。

Show 方法的功能是显示一个 UserForm 对象，具体语法如下。

```
object.Show modal
```

其中 object 代表 UserForm 对象，modal 代表窗体是模态的还是无模式的，当此参数赋值为 0 时表示是无模式的，赋值为 1 时表示是模态的。

当 UserForm 对象以模态方式运行时，不可以操作工作表和单元格，焦点总限制在 UserForm 对象之中；当 UserForm 对象以无模式方式运行时，焦点可以随意转移，允许在运行窗体的同时操作单元格或者工作表。

12.1.3　使用工具箱

工具箱中包含所有可用的控件，窗体中的一切控件都来自工具箱中。

1．工具箱与默认控件

工具箱外观如图 12-10 所示。



图 12–10　工具箱

工具箱中存放的常用控件如下。

Label 控件：即标签，图标为。

TextBox 控件：即文本框，图标为。

ComboBox 控件：即复合框，图标为。

ListBox 控件：即列表框，图标为。

CheckBox 控件：即复选框，图标为。

OptionButton 控件：即选项按钮，图标为。

ToggleButton 控件：即切换按钮，图标为。

Frame 控件：即框架，图标为。

CommandButton 控件：即命令按钮，图标为。

TabStrip 控件：即标签控件，图标为。

MultiPage 控件：即多页控件，图标为。

ScrollBar 控件：即滚动条，图标为。

SpinButton 控件：即微调按钮，图标为。

Image 控件：即图像控件，图标为。

RefEdit 控件：图标为。

2．添加其他控件到工具箱

工具箱中有 15 个默认的常用控件，也可以根据需要将其他控件添加到工具箱中。

其他控件有上百种，其中有一些功能比较强大的控件，例如 ListView（和列表框功能相近，但更美观更强大）、WebBrowser（网页控件）、ShockwaveFlash（Flash 动画播放控件）和 WindowsMediaPlayer（视频播放控件）等。

以 ShockwaveFlash 控件为例，在工具箱中添加新控件的步骤如下。

（1）在工具箱任意位置单击右键，从快捷菜单中选择“附加控件”菜单；

（2）在“附加控件”对话框中对“Shockwave Flash Object”打勾，然后单击“确定”按钮返回工具箱界面，在工具箱中将出现 ShockwaveFlash 控件的图标。

“附加控件”对话框如图 12-11 所示，添加到工具箱中的 ShockwaveFlash 控件图标如图 12-12 所示。

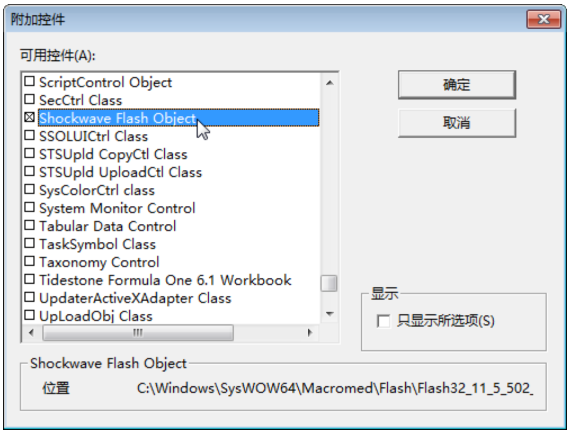


图 12-11　添加 ShockwaveFlash 控件



图 12-12　工具箱中的 ShockwaveFlash 控件

可以使用相同方法将 ListView、WebBrowser、和 WindowsMediaPlayer 控件添加到工具箱中，如图 12-13 所示。它们三者对应的名称如下。

ListView：Microsoft List Control,Version 6.0

WebBrowser：Microsoft Web Browser

WindowsMediaPlayer：Windows Media Player

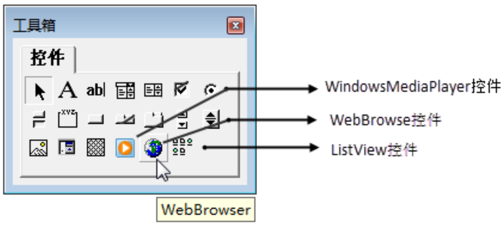


图 12-13　在工具箱中添加三种新的控件

3．将控件添加到窗体中

将工具箱中的控件添加到窗体中后才可以正常使用，添加方法如下。

（1）选择菜单“插入”→“窗体”。

（2）如果此时未显示工具箱，那么选择菜单“视图”→“工具箱”，从而打开工具箱。

（3）选择需要添加到窗体中的控件名称，例如 TextBox 控件。

（4）在窗体中单击，然后将鼠标光标向右下角拖曳，拖曳范围决定控件的宽度和高度，如图 12-14 所示。

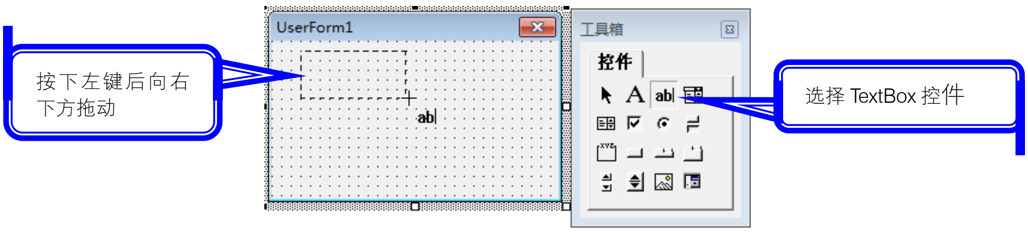


图 12-14　将工具箱中的 TextBox 控件添加到工具箱中

可以使用相同的方法将其他控件添加到窗体中。

12.1.4　标签控件

标签控件的类型名称是 Label。

在窗体中添加第一个标签时，默认的“名称”和“Caption”属性都是“Label1”，添加第二个标签时默认“名称”和默认“Caption”属性都是“Label2”，可以在代码中以“名称”属性引用此控件。

标签用于在窗体中添加说明性的文本，通常由开发者指定标签内容，该内容在窗体执行阶段不可手动修改。

在图 12-15 的 UserForm 对象中添加的一个标签控件，其名称由窗体对象左方的属性窗口中的“名称”属性决定，而显示的字符则由“Caption”属性决定。

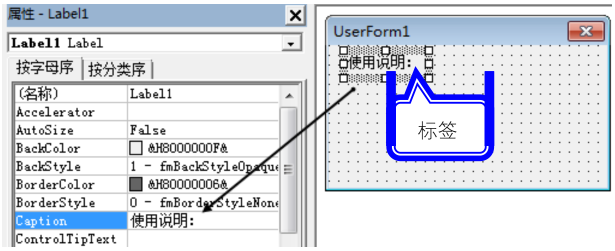


图 12–15　在窗体中添加标签并修改其显示文本

12.1.5　文本框控件

文本框控件的类型名称是 TextBox。

在窗体中添加文本框时，默认“名称”和“Caption”属性都是“TextBox1”，可用代码“TextBox1”引用此文本框。

文本框的用途是在运行窗体时让用户输入文字或者数值。在图 12-16 的窗体中包含一个标签和一个文本框控件。

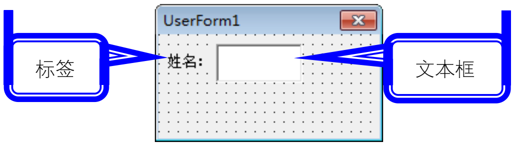


图 12–16　标签与文本框

12.1.6　命令控钮

命令按钮的类型名称是 CommandButton。

在窗体中添加命令按钮时，默认“名称”和“Caption”属性都是“CommandButton1”，可用代码“CommandButton1”引用此命令按钮。

命令按钮的用途是在单击按钮时可执行一个或者多个任务。在图 12-17 中，“确定”按钮的“名称”属性是“CommandButton1”，显示的文字是“确定”。

在 VBA 中引用图 12-17 的命令按钮应用“CommandButton1”，而非“确定”。

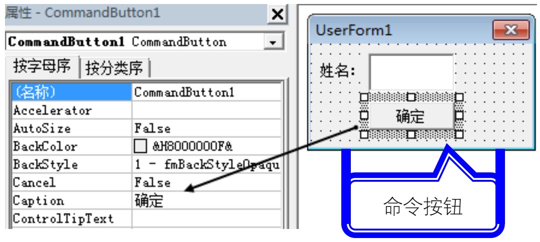


图 12–17 在窗体中添加命令按钮并修改其显示文本

12.1.7 复合框

复合框的类型名称是 ComboBox，也可称之为组合框。

可以将复合框理解为列表框与文本框的复合体，它既可以从列表中选择一个项目，也可以直接在文本框中输入数值。

图 12-18 中包含一个标签和一个复合框，在 VBE 界面中的复合框不会显示列表内容。

当运行窗体后，可以通过代码为复合框创建子项，单击复合框即可显示列表，而且可以根据需求显示单列或者多列。图 12-19 和图 12-20 即为运行状态下的复合框外观。

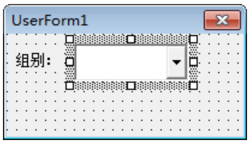


图 12–18 在窗体中添加复合框

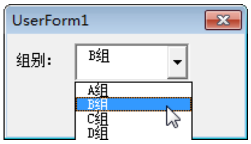
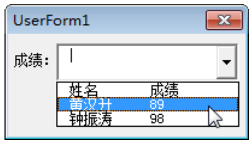


图 12–19 单列复合框图



12–20 双列复合框

本节仅展示复合框的外观，对于如何用代码为复合框创建子项，将在 12.2 节中详细介绍。

12.1.8 列表框

列表框的类型名称是 ListBox。

列表框可提供一个供用户选择的列表。它和复合框的差别有两点。

其一：列表框总是显示列表，而复合框默认状态只显示顶端的文本框，在单击文本框时才弹出下方的列表。

其二：列表框只允许从列表中选择，而复合框既允许选择也允许手动输入新值。

从工具箱中将列表框添加到窗体中时，其外观与文本框一致，如图 12-21 所示。不过当通过代码向列表框中添加数据并运行窗体后，列表框将与文本框大不相同，其可以同时显示多行多列的数据或者多行单列的数据，而且拥有普通样式和选项按钮样式两种外观。图 12-22 和图 12-23 分别展示了列表框的两种外观。

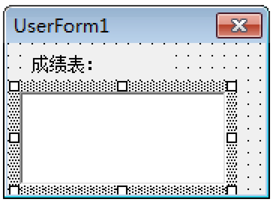


图 12–21 在窗体中添加列表框

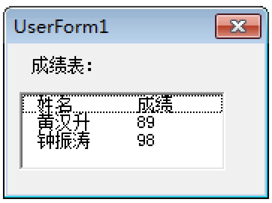


图 12–22 普通列表

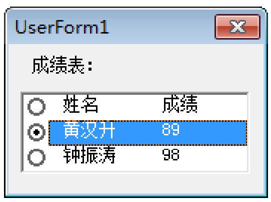


图 12–23 选项列表

在本章 12.2 节中将会介绍通过代码为列表框控件指定数据源的思路。

12.1.9　复选框

复选框的类型名称是 CheckBox。

复选框用于创建方框，此方框通过是否勾选来展示其所关联对象的状态。例如“视图”选项卡中的“网格线”即为复选框，勾选时表示显示网格线，没有勾选时表示不显示网格线。在工作中可以通过复选框来标示对象的某种状态。

在窗体中添加第一个复选框时，默认“名称”和“Caption”属性都是“CheckBox1”，在工作中使用时有必要修改其“Caption”属性。

图 12-24 的窗体中添加了 4 个复选框，每个复选框都采用了默认的“名称”属性，而“Caption”属性则分别修改为“音乐”、“运动”、“计算机”和“美食”。当窗体处于运行状态时，可以同时勾选多个复选框，如图 12-25 所示。

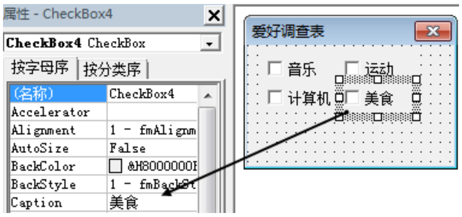


图 12–24　在窗体中添加 4 个复选框

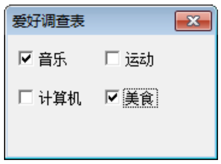


图 12–25　运行窗体时可以选择多个选项

12.1.10　选项按钮

选项按钮的类型名称是 OptionButton。

在使用选项按钮时通常会提供多个选项，但只允许用户从中选择一个项目。

在窗体中添加第一个选项按钮时，默认“名称”和“Caption”属性都是“OptionButton1”，因此在工作中使用选项按钮时有必要修改其“Caption”属性。

图 12-26 的窗体中添加了 4 个选项按钮，每个按项按钮都采用了默认的“名称”属性，而“Caption”属性则分别被修改为“音乐”、“运动”、“计算机”和“美食”。当窗体处于运行状态时，不可以同时选择多个选项按钮，如图 12-27 所示。

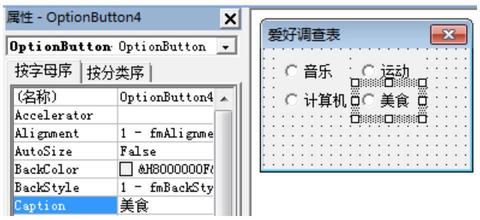


图 12–26　在窗体中添加 4 个选项按钮

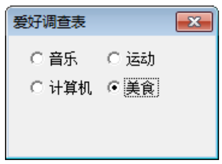


图 12–27　运行窗体时只能选择一个选项

12.1.11　框架

框架控件（也被称为分组框）的类型名称是 Frame。

框架有两个功能，其一是当窗体中控件太多时，利用框架控件将它们分组显示，从而美化窗体。其二是将选项按钮分组，突破其不能多选的限制。

在窗体中添加第一个框架时，默认“名称”和“Caption”属性都是“Frame1”，在工作中使用框架时有必要修改其“Caption”属性。

图 12-28 表示在窗体中添加第一个框架，其“Caption”属性已被修改为“音乐”。

图 12-29 是运行状态下的窗体，窗体中用 4 个框架将 8 个选项按钮分为 4 组，从而使组与组之间互不干扰，实现多选的目的。

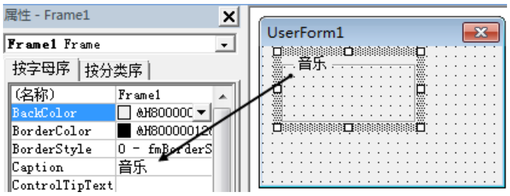


图 12–28　在窗体中添加框架



图 12–29　通过框架将选项按钮分组

12.1.12　切换按钮

切换按钮的类型名称是 ToggleButton。

切换按钮用于创建一个切换开关的按钮，类似于复选框，可以通过按钮的状态来表达它所关联对象的某种状态。例如在“开始”选项卡中的“格式刷”即为切换按钮，当按下该按钮时表示格式刷当前处于可用状态，再次单击按钮时表示格式刷处于不可用状态。

切换按钮有按下和弹起两种外观，在按下时，切换按钮的“Value”属性值为 False；在弹出时，切换按钮的“Value”属性值为 True。通常配合代码实现在按下与弹起时各执行不同的功能，同时让按钮在按下与弹起时显示不同的文本，即对“Caption”属性赋予不同的值。

图 12-30 表示在窗体中添加一个切换按钮和一个命令按钮，其中切换按钮的一切属性皆采用默认属性值，命令按钮则将默认的“Caption”属性值“CommandButton1”修改为“关闭窗口”。在设计模式下，命令按钮和切换按钮的外观一致。

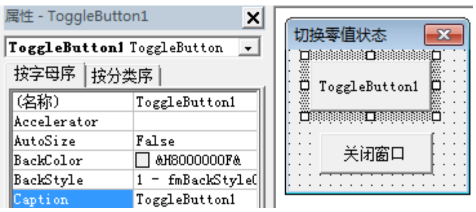


图 12–30　在窗体添加切换按钮和命令按钮

通过以下步骤可以实现切换按钮与工作表中的零值状态同步。

（1）双击窗体的空白区域进入窗体的代码窗口；

（2）删除自动产生的“UserForm_Click”事件过程代码，然后录入以下代码。

```
'显示窗体时执行，此过程为 UserForm 对象的 Activate 事件过程
Private Sub UserForm_Activate()                '放置位置：窗体的代码窗口中
 '让切换按钮的显示状态等于零值的显示状态（即显示零值时切换按钮为按下状态，否则为弹起状态）
    ToggleButton1.Value = ActiveWindow.DisplayZeros
    If ToggleButton1.Value Then                '如果切换按钮处于按下状态
        ToggleButton1.Caption = "显示零值"     '那么切换按钮所显示的文字为“显示零值”
    Else                                       '否则
        ToggleButton1.Caption = "不显示零值"   '切换按钮所显示的文字为“不显示零值”
    End If
End Sub
'单击切换按钮时执行，此过程为切换按钮的 Click 事件过程
Private Sub ToggleButton1_Click()
```

```
'让零值的显示状态等于切换按钮的显示态状（即切换按钮处于按下状态时显示零值，否则不显示）
    ActiveWindow.DisplayZeros = ToggleButton1.Value
    If ToggleButton1.Value Then                 '如果切换按钮处于按下状态
        ToggleButton1.Caption = "显示零值"       '那么切换按钮所显示的文字为“显示零值”
    Else                                        '否则
        ToggleButton1.Caption = "不显示零值"     '切换按钮所显示的文字为“不显示零值”
    End If
End Sub
'单击命令按钮时执行，此过程为命令按钮的 Click 事件过程
Private Sub CommandButton1_Click()
    Unload Me                                   '单击按钮时关闭当前窗体
End Sub
```

以上三个过程分别是 UserForm 对象的 Activate 事件、切换按钮的 Click 事件和命令按钮的 Click 事件。

其中“Unload Me”表示关闭当前窗体。Unload 方法表示关闭窗体，其语法如下。

```
Unload object
```

参数 object 代表窗体对象。

关键字 Me 在不同地方代表不同对象。如果出现在工作表事件中，Me 代表代码所在工作表的工作表对象；如果出现在工作簿事件中，Me 代表 ThisWorkbook，即代码所在工作簿；如果出现在窗体，Me 代表代码所在的窗体对象。

在本例中，将也可以“Unload Me”改为““Unload UserForm1”，表示关闭名为“UserForm1”的窗体。

（3）在代码窗口中按下【F5】键运行窗体，由于工作表中默认显示零值，所以窗体中的切换按钮将显示为按下状态，且显示的文字为“显示零值”，如图 12-31 所示。

（4）单击切换按钮，工作表中的零值马上处于隐藏状态，切换按钮同时呈现为弹起状态，如图 12-32 所示。

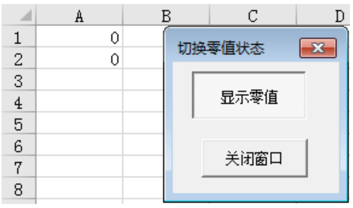


图 12–31　不显示零值时的切换按钮



图 12–32　显示零值时的切换按钮

本例案例文件请参考：..\ 第 12 章\ 12-3 通过切换按钮控制零值的显示方式.xlsm

12.1.13　多页控件

多页控件的类型名称是 MultiPage。

多页控件类似于框架，它们两者都可将有某种内在联系的多个控件单独作为一组显示。不过在显示方面稍有区别：多个分组框可以同时显示多个页面，但多页控件不能同时显示，就像工作表标签不能同时显示多个工作表一样。如果按钮较多，则采用多页控件将比采用框架控件更节约空间。

“设置单元格格式”对话框就是一个多页控件，在每个页面中存放了多个其他控件。

将多页控件添加到窗体中后，默认显示两页，分别为“Page1”和“Page2”。在实际工作中

需要修改它们的“Caption”属性，从而便于识别。

根据需要，可以删除页面或者添加新页，在页标题处单击右键，在快捷菜单中做出相应选择即可。

图 12-33 是默认状态下的多页控件，图 12-34 用于展示修改页标题文字的方法，图 12-35 则用于展示页标题的快捷菜单。

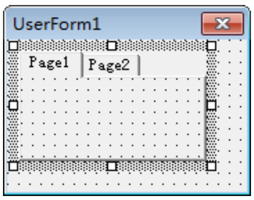


图 12–33　默认状态下的多页控件

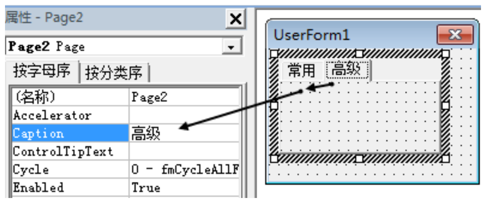


图 12–34　修改页标题文字



图 12–35　页标题的快捷菜单

12.1.14　滚动条

滚动条的类型名称是 ScrollBar，它和微调按钮的功能相近。

滚动条的功能是根据滚动块的位置，返回或设置另一控件的值。

滚动条通常用于设置另一个控件的数值，允许在一定范围中调整值的大小。例如设置窗口视图或者窗体的缩放比例，可以通过滚动条指定数据在 10 到 400 之间变化，当滚动条的值是 50 时，则用代码将窗口视图或者窗体的缩放比例调整为 50%，如图 12-36 所示。

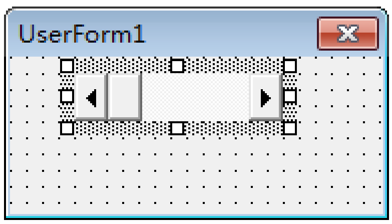


图 12–36　在窗体中添加一个滚动条

12.1.15　图像控件

图像控件的类型名称是 Image。

图像控件用于在窗体中显示图片，但不支持动画图片。

通常用图像控件为窗体进行装饰，或者对窗体中的列表内容做补注。例如在选择列表框中的某个值时，图像控件会马上加载与该值相关的图片。

12.1.16　Flash 控件

Flash 控件的类型名称是 ShockwaveFlash，它不是 Excel 自带的控件，在默认状态下不会显示在工具箱中，需要通过“附加控件”菜单将它添加到工具箱中。

Flash 控件用于显示动画，常用于在程序的帮助窗体中显示程序或者公司的 Logo 等。

在设计软件的“关于”或者“帮助”窗口时，通常会有软件的功能说明、官网地址和作者联系方式等。为了美观，可以在“关于”或者“帮助”窗口中插入一个 Flash 动画。

图 12-37 是一个简易的“Excel 百宝箱”软件帮助窗口，窗口顶部即为 Flash 动画，其“Movie”属性表示 Flash 动画的路径，指定路径后即可开始播放动画。

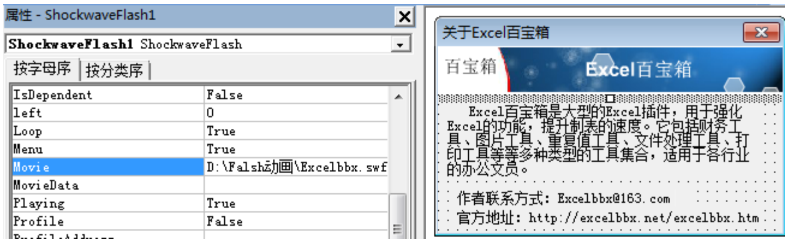


图 12–37　在窗体中插入 swf 格式的 Flash 动画

本例案例文件请参考：..\ 第 12 章\ 12-4 在窗体中播放 Flash 动画.xlsm

12.2　设置属性

在使用 UserForm 对象和添加 UserForm 对象时，需要对各种控件设置属性后才能使用，在默认状态下的 UserForm 对象或者控件没有实用性，更没有实际意义。设置属性有两种方法，即通过属性窗口设置和用代码对属性赋值，前者更直观，后者更强大。

本节重点介绍属性窗口以及常用控件的常用属性含义，其中部分无法通过属性窗口设置的属性则用代码完成。

12.2.1　属性窗口的用途

属性窗口可用于查看或者设置当前选中对象的大部分属性，某些只读属性不会显示在属性窗口中。也有一些属性虽然不是只读属性，但只能通过代码赋值，因此也不会显示在属性窗口中。

属性窗口可以显示或者设置工作表、ThisWorkbook、模块、类模块、UserForm 对象和控件的属性，不过本章仅介绍如何通过属性窗口设置 UserForm 对象和控件的属性。

当选中 UserForm 对象或者处于 UserForm 对象中的控件时，属性窗口中将显示此对象的属性值。在图 12-38 中选择了一个文本框控件，所以左方的属性窗口中可以看到此文本框的属性，例如“名称”、“AutoSize”、“AutoTab”、“AutoWordSelect”、“BackColor”和“BackStyle”等属性。可以随意修改这些属性值，例如将“BackStyle”属性由 fmBackStyleOpaque 修改为 fmBackStyleTransparent，那么此时文本框将成为透明背景。

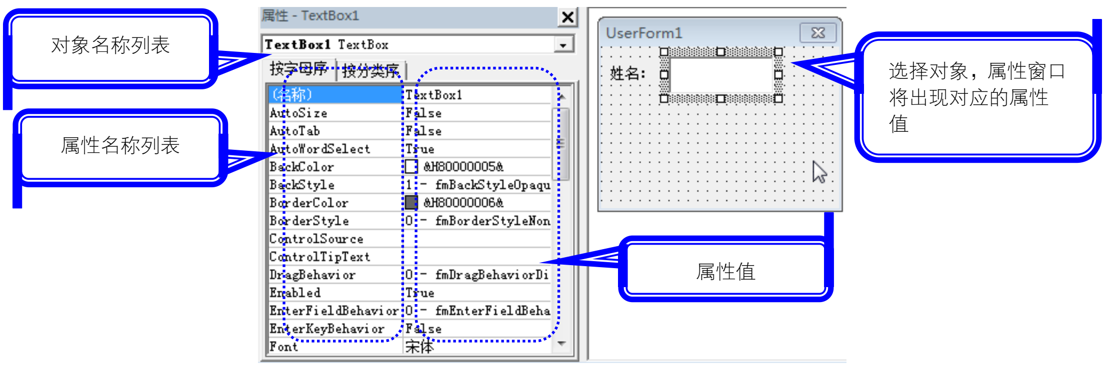


图 12–38　对象与对象的属性窗口

属性窗口的功能是展示属性值，以及接受修改属性的值。

一切显示在属性窗口中的属性都是既可读也可写的属性。

12.2.2 设置属性的两种方式

UserForm 对象或者控件的属性有两种设置方式，包括在属性窗口中设置和利用代码设置。

如果某个属性的值在运行窗体前就可以确定，而且在窗体的整个运行过程中不发生变化，那么此属性值适合通过属性窗口设置。例如文本框的“Left”、“Height”和“名称”等属性，命令按钮的“名称”、“Caption”和“Cancel”等属性。

如果某个属性满足以下条件之一，则需要利用代码为属性赋值。

1．属性值需要通过计算产生。

列表框中显示的值既可以是某个固定区域中的值，也可以是不固定区域的值。当用固定区域（例如“A1:C5”）作为列表框的数据源时，可以通过以下步骤设置列表框属性，使其可以显示 A1:C5 区域的值。

（1）选择菜单“插入”→“用户窗体”，然后将工具箱中的列表框控件拖到窗体中；

（2）选择列表框，将其“RowSource”属性设置为“A1:C5”，表示列表框的数据源是来自 A1:C5 区域的值；

（3）将列表框的“ColumnCount”属性设置为 3，表示列表框可以同时显示 3 列；

（4）将列表框的“ColumnWidths”属性设置为“40,40,40”，表示 3 列的列宽皆为 40，此时列表框显示效果如图 12-39 所示；

（5）选择窗体并按下【F5】键运行窗体，窗体显示效果如图 12-40 所示。

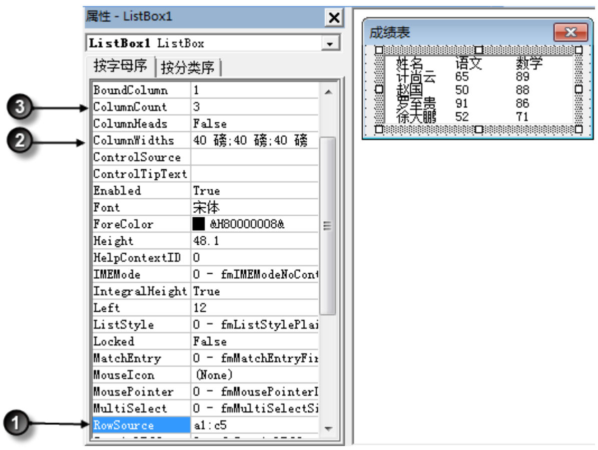


图 12–39　在属性窗口中设置列表框的属性

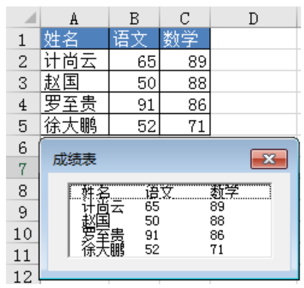


图 12–40　在窗体中显示 A1:C5 区域的值

如果将要求修改为“使用工作表的已用区域作为列表框的数据源”，由于设计窗体时不知道工作表的实际数据区域地址，那么需要利用代码“ActiveSheet.UsedRange.Address”计算而来，此时就不能在属性窗口对控件指定属性值。

完成“使用工作表的已用区域作为列表框的数据源”可按以下步骤实现：

（1）选择菜单“插入”→“用户窗体”，然后将工具箱中的列表框控件拖到窗体中；

（2）双击窗体任意部分进入窗体的代码窗口；

（3）删除自动产生的代码，然后重新录入以下代码。

```
'显示窗体时执行，此过程为 Userform 对象的 Activate 事件过程
Private Sub UserForm_Activate()                          '放置位置：窗体的代码窗口中
If IsEmpty(ActiveSheet.UsedRange) Then Exit Sub          '如果工作表是空表则退出程序
  Dim ColumnCount As Integer, RowCount As Integer        '声明两个变量用于保存已用区域的行列数
  ColumnCount = ActiveSheet.UsedRange.Columns.Count      '记录已用区域的列数
  RowCount = ActiveSheet.UsedRange.Rows.Count            '记录已用区域的行数
  ListBox1.Width = 40 * ColumnCount + 5                  '指定列表框的宽度为：每列 40，外加 5
  Me.Width = ListBox1.Width + 15                         '指定窗体的宽度为：列表框的宽度加 15
  ListBox1.Height = RowCount * 9                         '指定列表框的高度为：数据行数乘以 9
  Me.Height = ListBox1.Height + 35                       '指定窗体的高度为：列表框的高度加 35
  ListBox1.RowSource = ActiveSheet.UsedRange.Address     '指定列表框的数据源为已用区域地址
  ListBox1.ColumnCount = ColumnCount                     '指定列表的列数为：已用区域的列数
  '指定列表框的宽度：每列 40，赋值时“40,”的个数由已用区域的列数决定
  '由于 Rept 函数的计算结果最后一位是“,”，所以通过 Replace 函数将最后一位“，”删除
  ListBox1.ColumnWidths = Replace(WorksheetFunction.Rept("40,", ColumnCount) & ",", ",,", "")
End Sub
```

（4）在窗体的代码窗口中按下【F5】键运行窗体，窗体中的列表框将显示工作表中已用区域的所有值，如图 12-41 所示。

（5）工作表中添加新的数据，即扩展已用区域的行数与列数；

（6）再次运行窗体，窗体中的列表框将会自动调整宽度与高度，使其刚好适应新的数据区域，效果如图 12-42 所示。

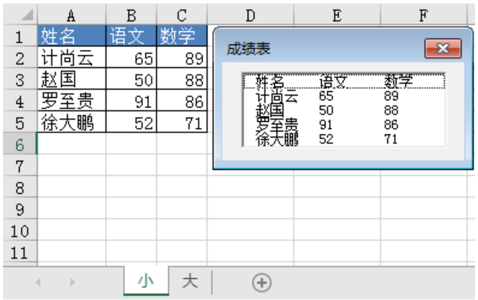


图 12-41　运行窗体后自动设置属性 1

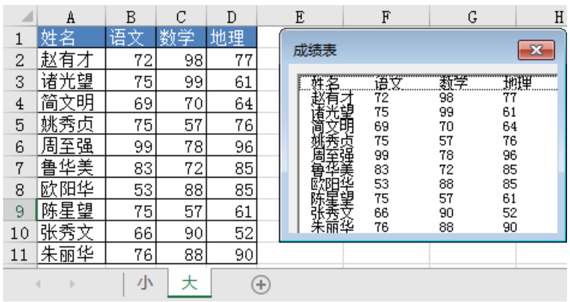


图 12-42　运行窗体后自动设置属性 2

很显然，利用代码给窗体或者控件的属性赋值将比直接在属性窗口中赋值灵活得多。

2．属性值会根据条件不断变化

控件的某些属性在窗体运行过程中会不断变化，这类属性只能通过代码来赋值。例如切换按

钮中的“Caption”属性即为典型的代表。在该案例中，切换按钮的“Caption”属性在不断变化，每单击一次按钮就会变化一次“Caption”属性值。

接下来将详细介绍 8 个常用控件的常用属性。

本例案例文件请参考：..\ 第 12 章\ 12-6 根据行列数自动为窗体与列表框的属性赋值.xlsm

12.2.3 文本框属性

在工作中文本框的使用较为频繁，文本框控件有 40 多种属性， VBA 初学者很难通过属性窗口中的属性名称明白每个属性的含义。

所以本小节将文本框控件的所有属性一并整理出来供读者参考，如表 12-1 所示。

表 12-1 文本框的属性含义与默认值说明

属性名称	含　义	可选值
名称	文本框的名称	不限，但需遵循与过程名称相同的命名规则（默认值：TextBox1）
AutoSize	是否自动调整大小以显示完整的内容	True False（默认值）
AutoTab	在文本框中输入的字符达到MaxLength属性的值时是否可自动跳格	True False（默认值）
AutoWordSelect	在设置为True时表示按下[Shift]键；在多选字符时以单词作为扩展选定内容的基本单元，赋值为False时表示以字符作为扩展选内容的基本单元	True（默认值） False
BackColor	设置文本框的背景颜色，可使用颜色编码，也可使用调色板设置颜色	Excel支持的所有颜色编码（默认值：&H80000005&）
BackStyle	设置文本框的背景方式，即是否显示为透明背景	fmBackStyleTransparent fmBackStyleOpaque（默认值）
BorderColor	设置文本框的边框颜色	Excel支持的所有颜色编码（默认值:&H80000006&）
BorderStyle	设置文本框的边框类型，即是否显示边框	mBorderStyleNone（默认值） fmBorderStyleSingle
ControlSource	指定Value属性的数据源，即单元格地址	单元格地址，A1、C5等（默认值：空文本）
ControlTipText	指定当用户将鼠标光针放在文本框上但未单击时所显示的文本	任意字符串（默认值:空文本）
DragBehavior	指定系统是否允许在文本框中执行拖放功能	fmDragBehaviorDisabled（默认值） fmDragBehaviorEnabled
Enabled	指定文本框能否接受焦点和响应用户产生的事件，即是否可用	True（默认值） False
EnterFieldBehavior	指定进入文本框时的选择行为,包括全选和选择上次所选的内容两种方式	fmEnterFieldBehaviorSelectAll（默认值） fmEnterFieldBehaviorRecallSelection
EnterKeyBehavior	用于指定按下Enter时创建新行还是将焦点移到下一个控件上	True False（默认值）
Font	用于设置文本框中的字体	当前系统中所有已安装的字体（默认值：宋体）
ForeColor	用于设置字体的颜色	Excel支持的所有颜色编码（默认值:&H80000008&）
Height	设置文本框的高度	大于0且小于窗体的高度（默认值是在添加控件时鼠标光标拖曳范围）可以更高，但在大于窗体的高度时不再有实际意义

继表

属性名称	含　　义	可选值
HelpContextID	为控件指定自定义的帮助文件中的特定主题	帮助文件中主题的上下文 ID号码（默认值：0）
HideSelection	指定当文本框没有焦点时被选定的文本是否保持突出显示	True（默认值） False
IMEMode	为文本框指定输入法编辑器 (IME) 的默认的运行时间模式。	0到11（默认值:0）
IntegralHeight	指定文本框是显示全部文本行还是显示部分行（不过实际测试结果此属性无效，不管如何设置都没有变化）	True（默认值） False
Left	设置文本框相对于窗体的左边距	不限（默认值由添加控件时的鼠标光标拖曳范围决定）
Locked	指定文本框能否被编辑	True False（默认值）
MaxLength	规定文本框中可输入的最多字符数	大于等于0的整数（默认值：0）
MouseIcon	为文本框指定一个自定义的鼠标图案，需配合MousePointer使用	所有图片文件，宜用ICO格式的图片（默认值：None）
MousePointer	指定用户把鼠标光标移到文本框上时所显示鼠标光标的类型，99表示显示自定义的图案	99和0到15的整数（默认值:0）
MultiLine	设置文本框中是否可以显示多行	True False（默认值）
PasswordChar	指定在文本框中是显示占位符还是显示实际输入的字符	任意字符（默认值：空文本）
ScrollBars	指定文本框是否有垂直或水平滚动条	0、1、2、3（默认值:0）
SelectionMargin	规定用户能否通过单击文本左边区域来选中一行	True（默认值） False
SpecialEffect	指定文本框的外观（主要针对边框样式）	0、1、2、3、6（默认值:2）
TabIndex	指定当前文本框在窗体【Tab】键顺序中的位置	从0到具有TabIndex属性的控件数减1之间（默认值由插入文本框时的顺序决定）
TabKeyBehavior	决定是否允许制表符出现在编辑区。MultiLine值为True时此属性才生效	True False（默认值）
TabStop	指定当用户跳格到当前文本框时，文本框能否获得焦点。赋值为False时表示不能通过按[Tab]键点定位到此文本框	True（默认值） False
Tag	存储对象的附加信息，可以简单理解为文本框的批注。	任意字符（默认值：空文本）
Text	设置文本框中显示的文本	任意字符（默认值：空文本）
TextAlign	定义控件中文本的对齐方式，包括左对齐、居中和右对齐	fmTextAlignLeft（默认值） fmTextAlignCenter fmTextAlignRight
Top	设置文本框相对于窗体的上边距	不限（默认值由添加控件时鼠标拖动范围决定）
Value	设置文本框中显示的文本，和Text属性同步	任意字符（默认值：空文本）
Visible	指定文本框是否可见	True（默认值） False
Width	设置文本框的宽度	大于0且小于窗体的宽度(默认值添加控件时鼠标拖动的范围决定） 可以更宽，但大于窗体的宽度时不再有实际意义
WordWrap	指出文本框的内容在行末是否自动换行（当MultiLine为False时，WordWrap被忽略）	True（默认值） False

其中有以下 9 个属性需要特别讲解。

1．根据内容调整宽度：AutoSize

此属性用于控制文本框是否自动调整大小以从而使输入内容完整显示。当赋值为 False 时表

示文本框显示为固定大小，赋值为 True 时则随输入文字的多少自动调整宽度。例如，输入“中国”，则文本框显示两个字符的宽度，输入“中国人民大会堂”，则自动调整为 7 个字的宽度。

2．指定最大长度：MaxLength

此属性表示文本框的最大长度，例如文本框用于输入身份证号，那么可以将此属性设置为 18，因为身份证号码不可能超过 18 位。如果文本框用于输入“性别”，则可以将此属性设置为 1，当输入一个字符后 VBA 将会阻止录入新的字符。

3．自动跳格：AutoTab

“AutoTab”属性需要与“MaxLength”属性共用才生效，当把“MaxLength”和“AutoTab”属性设置为 True 后，文本框中录入的字符达到最大长度时可以自动跳格，此功能十分有用。

例如窗体时有 10 个文本框，分别用于录入员工姓名(最大长度为 4)与工号(最大长度为 3)，那么可以将其中 5 个表示姓名的文本框的“MaxLength”属性设置为 4，将 5 个表示工号的文本框的“MaxLength”属性设置为 3，再将这 10 个文本框的“AutoTab”属性设置为 True。这样当在姓名文本框中录入 4 个字符后将会自动跳格到工号文本框中，在工号文本框中录入 3 位工号后又自动跳格到下一个姓名文本框中……如此设置可以大大节约操作时间，减少多次按【Enter】键或者使用鼠标单击的时间。

本例案例文件请参考：..\ 第 12 章\ 12-6 录入姓名与工号时自动跳格.xlsm

4．控制换行：MultiLine

此属性可以控制文本框是否允许换行，赋值为 True 时表示允许换行，默认值为 False。当文本框中的字符较多时有必要将此属性赋值为 True，从而使得当输入的字符达到右边的边界时可以自动切换到下一行继续录入。

5．控制换行还是跳格：EnterKeyBehavior

“EnterKeyBehavior”属性用于控制按下【Enter】键后的行为是换行还是跳格（将焦点转移到下一个控件上），此属性需要与“MultiLine”属性共用才生效。

在“MultiLine”属性和“EnterKeyBehavior”属性都设置为 True 时，按下【Enter】键可以在文本框内换行；在“MultiLine”属性设置为 True、“EnterKeyBehavior”属性设置为 False 时按下【Enter】键则会转移焦点，而非换行，此时只有按下【Ctrl+Enter】组合键才可以换行。

在图 12-43 中，第一个文本框可以按【Enter】键换行，输入“经商”后，按【Enter】键可继续录入第二行字符，而在第二个文本框中输入“音乐”并按【Enter】键后，将转移焦点到第一个文本框，而非重起一行继续录入字符。



图 12-43　EnterKeyBehavior 属性的效果演示

本例案例文件请参考：..\ 第 12 章\ 12-7 控制换行还是跳格.xlsm

6．指定数据源：ControlSource

此属性用于指定文本框的数据源，赋值为 A1 时表示在文本框中显示 A1 单元格的值。不过此属性不等同于公式，不会在更新 A1 单元格的值时同步更新。

7．添加屏幕提示：ControlTipText

此属性可为文本框添加屏幕提示，当鼠标光针移至文本框时可在屏幕上显示此属性的值，通常用于提示用户当前文本框的功能或者注意事项。

在图 12-44 中第二个文本框的“ControlTipText”属性赋值为“只能录入 4 位数值的工号”，所以运行窗体后，当把鼠标光标移至此文本框时可以看到提示信息，从而提升录入数据的准确度。

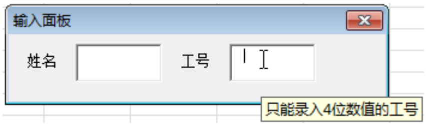


图 12-44　显示屏幕提示

8．设置 Tab 键顺序：TabIndex

此属性代表文本框的 Tab 键顺序。Tab 键顺序是指运行窗体后按下第几次【Tab】键时当前控件可获得焦点，该值从 0 开始，到具有“TabIndex”属性的控件总数量减 1 结束。

当运行窗体时，第一个具有焦点的控件的【Tab】键顺序值为 0，按下一次【Tab】键时可获得焦点的控件的“TabIndex”属性值为 1，再次按下【Tab】键时可获得焦点的控件的 TabIndex 属性值则为 2。所以可以设置调整文本框的“TabIndex”属性值来控制文本框的输入顺序。

假设窗体中有三个文本框，分别将它们的“TabIndex”属性值赋值为 0、2、1，那么运行窗体后默认焦点在第一个文本框，输入数据后按下【Tab】键则进入第三个文本框中，当第三个文本框输入数据后按下【Tab】键则进入第二个文本框中。此时若没有“TabIndex”属性值为 4 的文本框，则在按下【Tab】键后返回第一个文本框。

9．显示占位符：PasswordChar

此属性可以指定文本框中的占位符，也称掩码，即不显示原本输入的字符，而是在屏幕上显示为一个占位符。通常使用此属性来防止泄露录入的信息，例如录入密码。

如果需要文本框中输入字符时显示为星号，那么对文本框的“PasswordChar”属性赋值为“*”即可。

12.2.4　命令按钮属性

命令按钮有 32 个属性，比文本框的属性少很多，而且其中绝大部分属性与文本框的属性名称一致、属性功能一致，设置属性的方法也一致。所以本节仅介绍其中不重复而又比较有用的三个属性。

1．设置标题：Caption

此属性用于设置命令按钮的标题文字，通常是“确定”或者“取消”之类的按钮文字。

“Caption”属性可供用户识别、区分按钮的功能，多个命令按钮的“Caption”属性允许重

复。而命令按钮的“名称”属性是用于程序识别的，不允许多个命令按钮的“名称”属性重复，否则程序无法调用。

在图 12-45 中名为“CommandButton2”的命令按钮的“Caption”属性值为“取消”，所以在代码中调用按钮时用“CommandButton2”，而在窗体界面中显示的文本则是“取消”。为了叙述方便，通常称之为“取消”按钮，而实际上称之为“CommandButton2”按钮更精确。

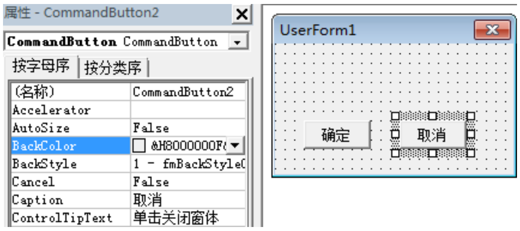


图 12–45　Caption 与名称属性

2．设置加速键：Accelerator

此属性用于指定命令按钮的加速键。例如“Accelerator”属性赋值为 A 时，那么可以按下【Alt+a】组合键执行此命令按钮。

在随书案例中文件中，窗体有两个命令按钮，第 1 个命令按钮用于新建工作表，其“Accelerator”属性赋值为 A，第 2 个命令按钮用于删除工作表，“Accelerator”属性赋值为 B。当运行窗体后可，按下【Alt+a】组合键可以新建工作表，按下【Alt+b】组合键则可以删除工作表。

本例案例文件请参考：..\ 第 12 章\ 12-8 指定命令按钮的加速键.xlsm

3．标识窗体的默认按钮：Cancel 与 Default

窗体中的默认按钮有两种，一是在按下【Esc】键时执行的命令按钮，在属性窗口中将命令按钮的“Cancel”属性设置为 True 即可；二是在其他命令按钮不具有焦点时，按下【Enter】键可执行的命令按钮，将命令按钮的“Default”属性设置为 True 即可。

在实际工作中尽量不要通过“Default”属性来设置默认按钮，使用“Cancel”属性即可。

在工作中，通常将关闭窗体的命令按钮设置为默认按钮，设置步骤如下。

（1）选择菜单“插入”→“用户窗体”，然后在窗体中添加两个命令按钮；

（2）将名为“CommandButton1”的命令按钮的“Caption”属性设置为“确定”，将名为“CommandButton2”的命令按钮的“Caption”属性设置为“取消”；

（3）将名为“CommandButton2”的命令按钮的“Cancel”属性设置为“True”；

（4）在窗体中双击名为“CommandButton2”的命令按钮，从而打开窗体的代码窗口，代码窗口中将自动出现以下代码。

```
Private Sub CommandButton2_Click()

End Sub
```

这是“CommandButton2”按钮的单击事件过程外壳。

（5）在单击事件的过程外壳中输入关闭窗体的代码“Unload Me”；

（6）在代码窗口中按下【F5】键运行窗体，然后按下【Esc】键，窗体将自动关闭。表明默认按钮可以通过快捷键调用。

本例案例文件请参考：..\ 第 12 章\ 12-9 设计按 Esc 键关闭的窗体.xlsm

12.2.5　复选框属性

复选框控件有 32 个属性，其中近 30 个属性和文本框控件一致。本小节介绍其中 3 个属性，包括“Picture”、“Caption”和“Value”属性。

在实际工作中，通常通过设置复选框的“Caption”属性来标识每个复选框所代表的对象，例如在图 12-46 中 6 个复选框分别使用了“钢琴”、“爬山”、“溜冰”、“唱歌”、“乒乓球”、“跳舞”作为“Caption”属性值。

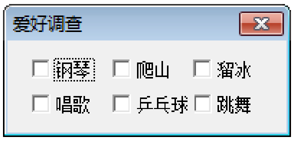


图 12-46　复选框的 Caption 属性

实际上也可以通过“Picture”属性使用图片来标识每个复选框的作用。以下案例展示了“Picture”属性与“Caption”属性搭配应用的思路。

（1）选择菜单“插入”→“用户窗体”，然后在窗体中添加 12 个复选框按钮和 2 个命令按钮；

（2）将窗体的“Caption”属性修改为“选择您喜欢的几种动物”，将两个命令按钮的“Caption”属性分别修改为“确定”与“取消”，然后将“取消”按钮的“Cancel”属性设置为 True，表示在按下【Esc】键时可以关闭窗体；

（3）将 12 个复选框的“Caption”属性分别修改为鼠、牛、虎、兔、羊、龙、蛇、马、猴、鸡、狗、猪，同时准备对应的 12 张动物的图片；

（4）选择第一个复选框，在属性窗口中单击“Picture”属性右方的浏览按钮（图标:），程序会弹出“加载图片”对话框，在对话框中选择对应的图片后返回窗体界面，窗体中的第一个复选框将不再显示文字“鼠”，而是显示与鼠对应的图片；

（5）重复步骤（4），为其他每个复选框设置“Picture”属性；

（6）双击“确定”命令按钮进入窗体的代码窗口，在代码窗口中录入以下两个事件过程：

```
'单击第 1 个按钮时执行，此过程为 CommandButton1 按钮的 Click 事件
Private Sub CommandButton1_Click()                    '放置位置：窗体的代码窗口中
    Dim ControlItem As Control, xlStr As String       '声明 2 个变量
    For Each ControlItem In Me.Controls               '遍历窗体中所有控件
        If ControlItem.Name Like "CheckBox*" Then      '如果控件的名字以“"CheckBox”开头
            '如果控件的“Value”属性值为 true，那么将它们的 Caption 属性与换行符串连起来
            If ControlItem.Value = True Then xlStr = xlStr & Chr(13) & ControlItem.Caption
        End If
    Next ControlItem
    MsgBox "您选择的是：" & xlStr                      '报告用户选择的对象名称
End Sub
```

以上过程表示在单击“确定”按钮时，循环检查窗体中每一个名字以“CheckBox”开头的控件，如果它的“Value”属性值为 True，那么将其“Caption”属性值与换行符串连起来，该字符串即为用户选择的对象名称。

针对以上过程需要补充 5 个知识点。

其一，过程中的代码“Me.Controls”表示当前窗体中的控件集合，可以利用 Control 型的变

量遍历这个控件集合，从而逐一获取每个子对象中的某些属性。

其二，代码中的 Like 是一个运算符，用于判断两个字符串是否相似，支持“?”和“*”两个通配符。本例中的“CheckBox*”表示以“CheckBox”开头、以任意长度的任意字符结尾的字符串。由于 12 个复选框的 Name（即“名称”属性）命名方式是“CheckBox1”、“CheckBox2”、“CheckBox3”……所以通过 Like 运算符可以从所有控件中单独挑出复选框，排除命令按钮。

其三，排除复选框以外的控件也可以改用 TypeName 函数替代 Like 运算符。TypeName 函数可以计算控件的类别名称，如果其类别名称不是“CheckBox”，那么将排除在外。所以也可以将代码“ControlItem.Name Like "CheckBox*"”修改为“TypeName(ControlItem) = "CheckBox"”。

其四，复选框的“Value”属性值为 True 时表示打勾状态，否则表示未打勾。

其五，在设置复选框的“Caption”与“Picture”属性值时，后者将覆盖前者，但是可以在代码中调用前者。

```
'单击第二个按钮时执行，此过程为 CommandButton2 按钮的 Click 事件
Private Sub CommandButton2_Click()
 Unload Me    '关闭窗体
End Sub
```

以上过程表示在单击“取消”按钮时，关闭窗体。

（7）在代码窗口中按下【F5】键运行窗体，在窗体中勾选第 4 个、第 5 个、第 7 个和第 11 个复选框，然后单击“确定”按钮将得到如图 12-47 所示结果。



图 12–47　复选框的 Picture 属性

本例案例文件请参考：..\ 第 12 章\ 12-10 复选框的 Caption 与 Picture 属性.xlsm

在本例中复选框较多，每个复选框需要设置两个属性，所以手动设置比较耗时，可以利用代码实现同等功能。假设在工作簿文件相同路径下的“12 生肖”文件夹中有 12 个图片，分别以数字 1 到 12 命名，对应鼠、牛、虎、兔、羊、龙、蛇、马、猴、鸡、狗、猪 12 个生肖，那么可以在窗体中添加复选框后，直接使用代码为 12 个复选框指定“Caption”与“Picture”属性，代码如下。

```
'显示窗体时执行，此过程是 UserForm 对象的 Activate 事件
 Private Sub UserForm_Activate()                    '放置位置：窗体的代码窗口中
    Dim ControlItem As Control, Arr, Num As String   '声明 3 个变量
  '生成一个包含 12 生肖的数组
    Arr = Array("鼠", "牛", "虎", "兔", "羊", "龙", "蛇", "马", "猪", "猴", "
鸡", "狗")
    For Each ControlItem In Me.Controls             '遍历窗体中所有控件
     If ControlItem.Name Like "CheckBox*" Then      '如果控件的名字以“"CheckBox”开头
```

```
'获取复选框的编号（将"CheckBox"替换成空文本后即为 1 开头的编号）
        Num = Replace(ControlItem.Name, "CheckBox", "")
'从数组 Arr 中取值作为复选框的"Caption"属性值，由于数组的下标是 0，所以需要用 Num-1
        ControlItem.Caption = Arr(Num - 1)
'为复选框指定图片，路径为工作簿所在路径下的"12 生肖"文件夹中，名为 Num & ".jpg"的文件
         ControlItem.Picture = LoadPicture(ThisWorkbook.Path & "\12 生肖\" & Num & ".jpg")
    End If
  Next ControlItem
End Sub
```

本例案例文件请参考：..\ 第 12 章\ 12-10 复选框的 Caption 与 Picture 属性 2.xlsm

选项按钮与复选框的各项属性一致，此处不再详述。

12.2.6　列表框属性

列表框控件在属性窗口中包含 45 个属性，其中 37 个属性可在属性窗口中修改，其他属性要么是只读属性不能显示在属性窗口中，要么只能通过代码修改。本小节主要介绍以下 9 个属性的作用与使用方法。

1．设置列表框的数据源：RowSource

此属性用于指定列表框的数据源，属性值只能是区域地址，例如"A1:C10"。

尽量使用代码设置"RowSource"属性，因为这样比在属性窗口中设置灵活得多。通常可以在 UserForm 对象的 Activate 事件中设置列表框的属性，例如以下代码表示在显示窗体时为列表框指定数据源。

```
Private Sub UserForm_Activate()          '放置位置：窗体的代码窗口中
   ListBox1.RowSource = "A1:C10"         '指定列表框的数据源
End Sub
```

2．指定列表框的列数：ColumnCount

不管对"RowSource"属性赋值的区域包含多少列，列表框默认只能显示 1 列。当需要列表框显示多列时，必须指定"ColumnCount"值，因此需要在"RowSource"属性赋值后需要加入以下代码，否则列表框只显示左边一列数据：

```
ListBox1.ColumnCount = Range("A1:C10").Columns.Count
```

3．设置每一列的列宽：ColumnWidths

列表框不会根据每一列的内容自动调整每一列的宽度，需要在 UserForm 对象的 Activate 事件中指定每一列的列宽。"ColumnWidths"属性的格式为"第一列宽度,第二列宽度,第三列宽度……"

例如在列表框的数据源为"A1:C10"时，可采用"40,50,50"或者"20,100,100"作为"ColumnWidths"属性的值，具体由区域中的数据长度决定。

假设需要将图 12-48 中的A1:C6 区域显示在列表框中，那么可以在UserForm 对象的Activate事件中写入以下代码。

```
'显示窗体时执行，此过程是 UserForm 对象的 Activate 事件
Private Sub UserForm_Activate()          '放置位置：窗体的代码窗口中
  Me.ListBox1.RowSource = "A1:C6"        '指定列表框的数据源
  ListBox1.ColumnCount = Range("A1:C6").Columns.Count '由数据源的列数决定列表框的列数
```

```
    ListBox1.ColumnWidths = "40,30,30"    '指定每一列的宽度
End Sub
```

代码表示用 A1:C6 区域作为列表框的数据源，用数据源的列数决定列表框的列数，同时根据每列数据的宽度手动指定列表框中每一列的宽度。最终效果如图 12-48 所示。

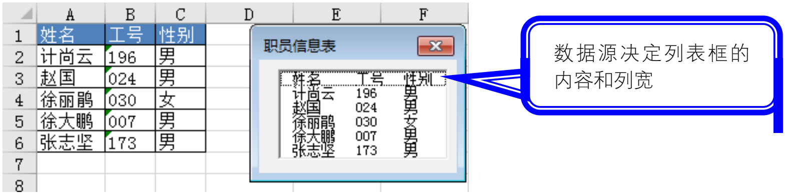


图 12–48　在列表框中显示 A1:C6 的值

本例案例文件请参考：..\ 第 12 章\ 12-11 列表框的 RowSource、ColumnCount 与 ColumnWidths 属性.xlsm

4．添加表头：ColumnHeads

此属性可以让列表框显示表头，从而使列表框显得更美观。

当列表框的“ColumnHeads”属性设置为 True 时，列表框的表头来自“RowSource”属性值的上一行的值，如果 RowSource”属性值本身就是从第一行开始，那么表头将显示为工作表的列标题。如果上一个案例的列表框需要显示表头，那么可按照以下方式设置列表框属性。

```
'显示窗体时执行，此过程是 UserForm 对象的 Activate 事件
Private Sub UserForm_Activate()                          '放置位置：窗体的代码窗口中
    ListBox1.RowSource = "A2:C6"                         '指定列表框的数据源
    ListBox1.ColumnHeads = True                          '让列表框显示表头
    ListBox1.ColumnCount = Range("A2:C6").Columns.Count  '由数据源的列数决定列表框的列数
    ListBox1.ColumnWidths = "40,30,30"                   '指定每一列的宽度
End Sub
```

通过以上设置，运行窗体将得到如图 12-49 所示的效果。

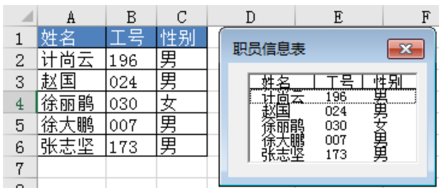


图 12–49　为列表框添加表头

本例案例文件请参考：..\ 第 12 章\ 12-12 列表框的 ColumnHeads 属性.xlsm

5．指定列表框的值：BoundColumn 与 TextColumn

“BoundColumn”与“TextColumn”属性分别表示列表框的“Value”属性和“Text”属性值在当前选中行中的顺序。换言之，当选择列表框中的某行时，列表框的“Value”属性值来自“BoundColumn”所指定的列，“Text”属性值来自“TextColumn”所指定的列。

接下来在上一个案例的基础上略作修改，说明 BoundColumn、TextColumn 与 Value、Text 属性之间的关系。

（1）将案例中列表框的“BoundColumn”和“TextColumn”两个属性分别改为 2 和 1；

（2）在窗体中插入两个选项按钮，并且将“Caption”属性修改为“姓名”和“工号”；

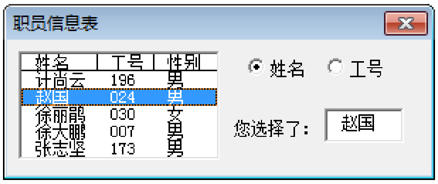
（3）在窗体中插入一个标签，将其“Caption”属性修改为“您选择了：”；

（4）在标签后面插入一个文本框；

（5）双击文本框，并输入以下代码。

```
'单击列表框时执行，此过程是列表框的 Click 事件过程
Private Sub ListBox1_Click()                    '放置位置：窗体的代码窗口中
    If Me.OptionButton1 Then                    '如果选择第一个选项按钮
        Me.TextBox1.Value = ListBox1.Text       '在文本框中显示列表框的 Text 属性值
    Else                                        '否则
        Me.TextBox1.Value = ListBox1.Value      '在文本框中显示列表框的 Value 属性值
    End If
End Sub
```

以上代码表示在选择“姓名”选项按钮后，单击列表框可将选中行的第 1 列的值显示在文本框中；如果选择“工号”选项按钮后单击列表框，可将选中行的第 2 列的值显示在文本框中，效果分别如图 12-50 和图 12-51 所示。



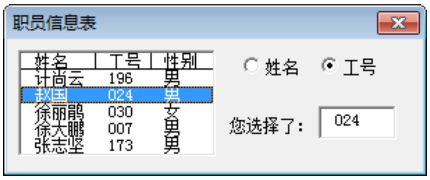


图 12–50　获取列表框的 TextColumn 值　图 12–51　获取列表框的 BoundColumn 值

本例案例文件请参考：..\ 第 12 章\ 12-13 指定列表框的值所在的列.xlsm

6．设置列表框外观：ListStyle

“ListStyle”属性决定列表框的外观是普通样式还是选项按钮样式。当“ListStyle”属性设置为“fmListStylePlain”时表示普通样式，图 12-49 的列表框即为普通样式外观。

如果将属性值设置为“fmListStyleOption”，那么将得到如图 12-52 所示的样式。

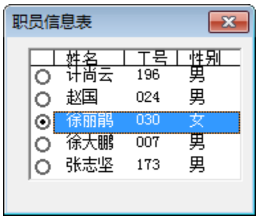


图 12–52　选项按钮样式的列表框

7．多选与选择状态：MultiSelect 与 Selected

“MultiSelect”属性用于控制列表框是否支持多选，当赋值为 True 时表示可以多选。

“Selected”属性则表示列表框中的每个值是否已被选中，值为 True 时表示当前行处于选中状态。

以下案例通过列表框的 AddItem 方法向第一个列表框中添加工作簿中所有工作表的名称，然后允许用户将第一个列表框中的某些值转移到第二个列表框中，操作步骤如下。

（1）选择菜单“插入”→“用户窗体”，然后在窗体中添加两个列表框。

（2）在属性窗口中将第一个列表框的“MultiSelect”属性设置为 True，表示允许多项选择，并将“ListStyle”属性设置为“fmListStyleOption”，表示显示为复选框样式（在列表框支持多选时，“fmListStyleOption”表示显示复选框外观；在列表框不支持多选时，“fmListStyleOption”表示显示选项按钮外观）。

（3）在两个列表框中间插入一个命令按钮，将其“Caption”属性设置为“->”，此窗体效果如图 12-53 所示。

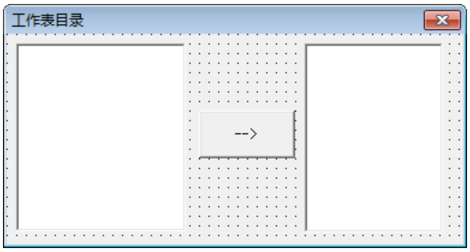


图 12-53　窗体控件布局

（4）双击窗体，进入窗体的代码窗口，删除自动产生的代码，然后录入以下新的代码。

```
'显示窗体时执行，此过程是 UserForm 对象的 Activate 事件过程
Private Sub UserForm_Activate()           '放置位置：窗体的代码窗口中
    Dim sht As Worksheet                  '声明 Worksheet 型的对象变量
    For Each sht In Worksheets            '遍历所有工作表
        ListBox1.AddItem sht.Name         '利用 AddItem 方法将工作表名称添加到列表框中
    Next sht
End Sub
```

此过程表示在显示窗体时将所有工作表名称添加到第一个列表框中。

代码中的 AddItem 方法表示向列表框中添加子项，一次只能添加一项。

与 AddItem 方法相对应的是 RemoveItem 方法，该方法可以根据需要删除指定的项目。

```
'单击命令按钮时执行，此过程是 CommandButton1 对象的 Click 事件过程
Private Sub CommandButton1_Click()                  '放置位置：窗体的代码窗口中
    Dim i As Byte, j As Byte                        '声明一个 Byte 型变量
    For i = 0 To Me.ListBox1.ListCount - 1          '遍历列表框的所有子项（下标为 0）
        If ListBox1.Selected(i) Then                '如果第 i 个值处于选中状态
            '如果第二个列表框已经有数据
            If ListBox2.ListCount > 0 Then
                For j = 0 To ListBox2.ListCount - 1     '遍历第二个列表框的所有子项
                    '如果第二个列表框中第 j 个项目与第一个列表框第 i 个项目相等
                    If Me.ListBox2.List(j) = ListBox1.List(i) Then
                        '提示用户已经重复，然后退出循环
                        MsgBox ListBox1.List(i) & "已存在，不要重复添加"
                        Exit For
                    End If
                    '如果变量 j 等于第二个列表框的项目数量减 1（即未发现重复者）
                    '那么将第一个列表框中第 i 个值添加到第二个列表框中
                    If J = ListBox2.ListCount - 1 Then ListBox2.AddItem
ListBox1.List(i)
                Next j
            Else    '否则（表示第二个列表框是空白的）
                ListBox2.AddItem ListBox1.List(i)   '将该值添加到第二个列表框中
```

```
                End If
            End If
        Next i
End Sub
```

此过程表示在单击命令按钮时，将第一个列表框中处于选中状态的项目添加到第二个列表框中，在添加前需要检查是否存在重复项，如果重复则提示用户，并且不再添加。

代码中的“ListCount”属性表示列表框中数据的行数可读不可写。“Selected”属性表示列表框项目是否处于选中状态，当值为 True 时表示选中状态，语法如下。

```
object.Selected( index ) [= Boolean]
```

其中 object 代表列表框，参数 index 代表索引号，从 0 开始到列表框总行号减 1 结束。

（5）在代码窗口中按下【F5】键运行窗体，在窗体中第一个列表框显示活动工作簿中所有工作表的名称；

（6）任意选择列表中的多个项目，然后单击“→”按钮，将在第二个列表框中出现第一个列表框中选中的项目，效果如图 12-54 所示。

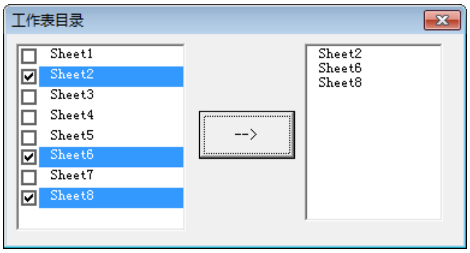


图 12-54　将第一个列表框项目添加到第二个列表框中

本例案例文件请参考：..\ 第 12 章\ 12-14 列表框的 MultiSelect 与 Selected 属性.xlsm

8．设置列表框的列表条目：List

使用列表框引用单个区域中的值时，在属性窗口中设置“RowSource”属性即可，若引用多个区域或者引用不确定区域的值时，则只能使用代码提取目标数据，然后将数据导入到数组变量中，再将数组变量赋值给列表框的“List”属性。“List”属性不在属性窗口中，只能用代码赋值。

假设要求在窗体中显示产量表中所有大于 500 的产量信息，由于哪些行的产量大于 500 只能通过计算获得，而且大于 500 的产量信息并不都是连续出现的，所以只能通过以下步骤完成。

（1）选择菜单“插入”→“用户窗体”，然后在窗体中添加 1 个列表框。

（2）双击窗体进入窗体的代码窗口，删除自动产生的代码，然后录入以下新的代码。

```
'单击命令按钮时执行，此过程是 UserForm 对象的 Activate 事件过程
Private Sub UserForm_Activate()          '放置位置：窗体的代码窗口中
  Dim Arr, RowItem As Integer '声明两个变量，前者用于保存产量数据，后者用于循环语句的变量
'声明两个变量，前者用于保存符合条件的所有信息，后者代表符合条件的数量
  Dim Arr2(), TargetCount As Integer
  Arr = ActiveSheet.UsedRange.Value      '将已用区域的值赋予变量 Arr
  For RowItem = 1 To UBound(Arr)         '遍历数组 Arr 的每一行
  If Arr(RowItem, 4) > 500 Then          '如果大于 500（标题“产量”也大于 500）
    TargetCount = TargetCount + 1        '统计符合条件的产量个数
  '根据符合条件的产量个数重置数组 Arr2 的上标
    ReDim Preserve Arr2(1 To 4, 1 To TargetCount)
    Arr2(1, TargetCount) = Arr(RowItem, 1) '将符合条件的 Arr 中的四列值都导入到数组 Arr2 中
```

```
            Arr2(2, TargetCount) = Arr(RowItem, 2)
            Arr2(3, TargetCount) = Arr(RowItem, 3)
            Arr2(4, TargetCount) = Arr(RowItem, 4)
        End If
        Next RowItem
        Me.ListBox1.ColumnCount = 4                    '让列表框显示为四列
         Me.ListBox1.ColumnWidths = "40,30,30,30"      '分别指定四列的宽度
    '将数组 Arr2 转置方向，然后赋值给列表框的 List 属性，从而使数组的值显示在列表框中
         Me.ListBox1.List = WorksheetFunction.Transpose(Arr2)
    End Sub
```

以上代码表示将“产量表”的已用区域的值转换成数组，然后通过循环语句判断数组中第四列的值是否大于 500。如果大于 500 则将该行数据导入到数组变量 Arr2 中，在导入完成后将数组 Arr2 赋值给列表框的“List”属性，同时设置列表框的“ColumnCount”和“ColumnWidths”属性，确保列表框可以将信息完整显示出来。

（3）在代码窗口中按下【F5】键运行窗体，窗体中的列表框将显示“产量表”中所有产量大于 500 的信息，如图 12-55 所示。

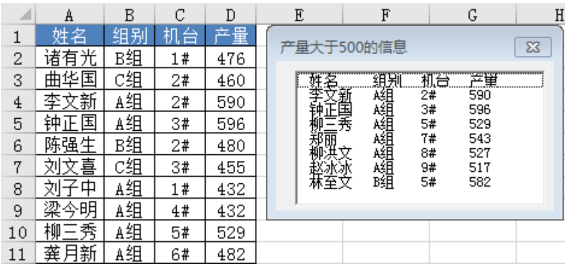


图 12–55　提取工作表中产量大于 500 的信息并显示在列表框中

AddItem 方法一次只能添加单个值，而本例需要显示四列内容，所以不能使用 AddItem 方法逐一向列表框添加内容，而是借助数组变量作为中转站，最后将数组的值导入列表框。

本例案例文件请参考：..\ 第 12 章\ 12-15 在列表框中显示产量大于 500 的信息.xlsm

12.2.7　复合框属性

复合框（组合框）有 62 个属性，其中 50 个属性显示在属性窗口中，其余可读不可写的属性以及只能通过代码赋值的属性则不会显示在属性窗口中，可以通过关键字“组合框控件”在帮助中找到这些属性。

复合框控件与列表框控件的绝大部分属性相同，而且常用属性 BoundColumn、ColumnCount、ColumnHeads、ColumnWidths、TextColumn、List、ListStyle、RowSource、Value 和 Text 等都一致，所以本节主要介绍复合框专用的两个属性：ListWidth 和 Style。

1．指定复合框宽度：Width、ColumnWidths 和 ListWidth

“Width”属性表示复合框控件在窗体中占用的宽度，“ColumnWidths”表示复合框列表的每一列宽度，“ListWidth”代表列表的整体宽度。当复合框列表包含多列时，这三个属性都有必要设置。以下案例需要用到这三个属性。

假设工作簿中有三个工作表，分别为“一班”、“二班”和“三班”，每个工作表中包含该班的学生姓名和成绩，要求在窗体中查询成绩“不及格”、“及格”、“良好”和“优秀”4 类的学生

信息，其中“不及格”代表小于 60 分的成绩，“及格”代表大于等于 60 并且小于 70 分的成绩，“良好”代表大于等于 70 分并且小于 90 分的成绩，“优秀”代表大于等 90 分并且小于等于 100 分的成绩。

实现以上需求的操作步骤如下。

（1）选择菜单“插入”→“用户窗体”，然后在窗体中添加一个标签控件、一个复合框和一个列表框，并按照如图 12-56 所示的方式排列。

（2）双击窗体进入窗体的代码窗口，删除自动产生的代码，然后录入以下新的代码。

```
'显示窗体时执行，此过程是 UserForm 对象的 Activate 事件过程
Private Sub UserForm_Activate()                    '放置位置：窗体的代码窗口中
    '将四行二列的数组赋值给复合框的 List 属性
    ComboBox1.List = [{"不及格","60 分以下";"及格","60 到 70 分";"良好","70 到 90 分";"优秀","90 分以下"}]
    ComboBox1.ColumnCount = 2                      '让复合框同时显示 2 列
    ComboBox1.Width = 50                           '复合框的宽度为 50
    ComboBox1.ColumnWidths = "50,60"               '复合框的第一列宽度 50，第二列宽度 60
    ComboBox1.ListWidth = 110                      '复合框的列表宽度为 110
End Sub
```

以上代码表示在显示窗体时将一个二维数组赋值给复合框的“List”属性，然后根据数组的特点设置复合框的列数、宽度、每列列宽和列表整体宽度。

```
'修改复合框的值时执行，此过程是 ComboBox1 对象的 Change 事件过程
 Private Sub ComboBox1_Change()
    Dim Arr, RowItem As Integer, sht As Worksheet  '声明变量
    Dim Arr2(), TatgetCount As Integer             '声明变量
    TatgetCount = 1                                '指定初始值为 1
    ReDim Preserve Arr2(1 To 3, 1 To TatgetCount)  '根据 TatgetCount 的值重置 Arr2 的上标
    Arr2(1, TatgetCount) = "班级"                  '写入第一个标题
    Arr2(2, TatgetCount) = "姓名"                  '写入第二个标题
    Arr2(3, TatgetCount) = "成绩"                  '写入第三个标题
For Each sht In Worksheets                         '遍历所有工作表
     '将工作表的 A1 单元格的当前区域的值赋予变量 Arr
     Arr = sht.Range("A1").CurrentRegion.Value
     For RowItem = 1 To UBound(Arr)                '遍历数组 Arr 的每一行
            Select Case ComboBox1.Text             '根据复合框的值决定查找方式
            Case "不及格"                          '假设复合框的值是“不及格”
                 '如果数组 Arr 第二列的值（即成绩）大于等于 0 而且小于 60
                 If Arr(RowItem, 2) >= 0 And Arr(RowItem, 2) < 60 Then
                        TatgetCount = TatgetCount + 1          '累加计数器
                      '根据 TatgetCount 的值重置变量 Arr2 的上标
                      ReDim Preserve Arr2(1 To 3, 1 To TatgetCount)
                      Arr2(1, TatgetCount) = sht.Name          '将工作表名称写入数组中
                      Arr2(2, TatgetCount) = Arr(RowItem, 1)   '将姓名写入数组
                      Arr2(3, TatgetCount) = Arr(RowItem, 2)   '将成绩写入数组
                 End If
             Case "及格"                                       '假设复合框的值是“不及格”
                 '如果数组 Arr 第二列的值（即成绩）大于等于 60 而且小于 70
                 If Arr(RowItem, 2) >= 60 And Arr(RowItem, 2) < 70 Then
                     TatgetCount = TatgetCount + 1             '累加计数器
                     '根据 TatgetCount 的值重置变量 Arr2 的上标
```

```
                ReDim Preserve Arr2(1 To 3, 1 To TatgetCount)
                Arr2(1, TatgetCount) = sht.Name         '将工作表名称写入数组中
                Arr2(2, TatgetCount) = Arr(RowItem, 1)  '将姓名写入数组
                Arr2(3, TatgetCount) = Arr(RowItem, 2)  '将成绩写入数组
            End If
        Case "良好"
            '如果数组 Arr 第二列的值（即成绩）大于等于 70 而且小于 90
            If Arr(RowItem, 2) >= 70 And Arr(RowItem, 2) < 90 Then
                TatgetCount = TatgetCount + 1           '累加计数器
                '根据 TatgetCount 的值重置变量 Arr2 的上标
                ReDim Preserve Arr2(1 To 3, 1 To TatgetCount)
                Arr2(1, TatgetCount) = sht.Name         '将工作表名称写入数组中
                Arr2(2, TatgetCount) = Arr(RowItem, 1)  '将姓名写入数组
                Arr2(3, TatgetCount) = Arr(RowItem, 2)  '将成绩写入数组
            End If
        Case Else                                       '否则（表示复合框的值是“优秀”）
            '如果数组 Arr 第二列的值（即成绩）大于等于 90 而且小于等于 100
            If Arr(RowItem, 2) >= 90 And Arr(RowItem, 2) <= 100 Then
                TatgetCount = TatgetCount + 1           '累加计数器
                ReDim Preserve Arr2(1 To 3, 1 To TatgetCount)
                Arr2(1, TatgetCount) = sht.Name         '将工作表名称写入数组
                Arr2(2, TatgetCount) = Arr(RowItem, 1)  '将姓名写入数组
                Arr2(3, TatgetCount) = Arr(RowItem, 2)  '将成绩写入数组
            End If
        End Select
    Next RowItem
  Next sht
  If UBound(Arr2) > 1 Then                              '如果数组 Arr 的行数大于 1
    ListBox1.ColumnCount = 3                            '将列表框的列数设置为 3
    ListBox1.ColumnWidths = "40,40,40"                  '将每列的宽度设置这 40
    ListBox1.List = WorksheetFunction.Transpose(Arr2)   '将数组 Arr2 赋予列表框的 List 属性
  Else
    MsgBox "没有符合条件的成绩", vbOKOnly, "友情提示"否则提示用户
  End If
End Sub
```

以上代码表示在修改复合框的值时在所有工作表中查找复合框所指定的条件，然后将查找到的目标存放在到数组变量中，并且将数组变量赋值给列表框。

（3）在代码窗口中按下【F5】键运行窗体，单击复合框可弹出如图 12-57 所示的复合框。

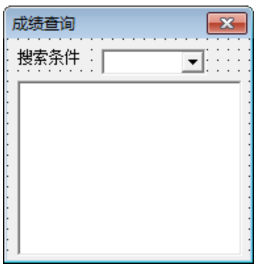


图 12–56　窗体控件布局

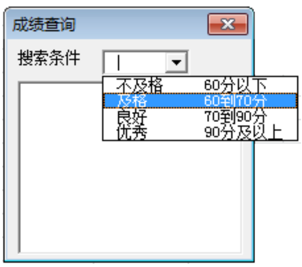


图 12–57　在复合框中显示查询条件

查询通常使用类似于数据有效性或者筛选功能的下拉列表，不过有效性和筛选都只能显示单列的下拉列表，而改用复合框则可以显示多列，可以将多列中的某一列设置为主键，其他列用于

对主键的值添加说明。例如本例的复合框显示两列，主键是分段名称，第二列则对主键加以描述，解释分类的范围，从而提升可读性。

（4）从复合框的列表中选择“及格”，列表框中将出现三个工作表中的大于等于 60 并且小于 70 这个范围的成绩信息，如图 12-58 所示。

（5）从复合框的列表中选择“优秀”，列表框中将出现三个工作表中的大于等于 90 并且小于等于 100 这个范围的成绩信息，效果如图 12-59 所示。

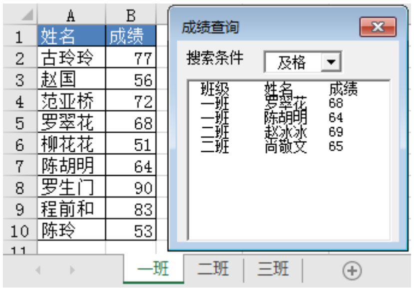


图 12-58　查询“及格”学生成绩

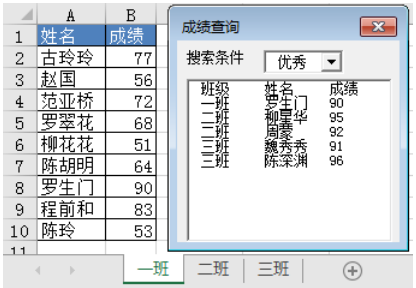


图 12-59　查询“优秀”学生成绩

在本例中，除用到复合框的几个属性外，也用到复合框的 Change 事件，以及 UserForm 对象的 Activate 事件。本书 12.3 节将会详细分析窗体与控件的各种事件。

本例案例文件请参考：..\ 第 12 章\ 12-16 复合框的 Width、ColumnWidths 和 ListWidth 属性.xlsm

2．确定是否允许录入值：Style

在上一个案例中，在“ComboBox1_Change”事件过程表示在修改复合框的值时执行查找，由于复合框既接受选择又接受直接输入字符，在输入字符不符合规范时将无法查到目标，甚至可能造成程序出错。复合框提供了“Style”属性来解决这个问题。

“Style”属性用于控制复合框是否允许手动输入值，当赋值为 fmStyleDropDownCombo 时，表示既可以选择下拉列表也可以手动输入值，当赋值为 fmStyleDropDownList 时表示只能从下拉列表中选择值，而不能手动输入。

所以，针对上一个案例，将复合框的“Style”属性设置为 fmStyleDropDownList，即可禁止用户手动录入字符，从而避免出错。

12.2.8　图像控件属性

图像控件包含 23 个属性，其中绝大多数属性与文本框控件的属性一致。仅有 3 个属性需要特意说明，分别是“Picture”、“PictureAlignment”和“PictureSizeMode”。

“Picture”属性表示图像控件的图片路径。

“PictureAlignment”属性代表图片的对齐方式，包括 fmPictureAlignmentTopLeft（左上角对齐）、fmPictureAlignmentTopRight（右上角对齐）、fmPictureAlignmentCenter（中心对齐）、fmPictureAlignmentBottomLeft（左下角对齐）和 fmPictureAlignmentBottomRight（右下角对齐）。

“PictureSizeMode”属性用于控制背景图片的显示方式，包括 fmPictureSizeModeClip（裁掉图片中比窗体或页面大的部分，是默认的显示方式）、fmPictureSizeModeStretch（扩展图片使其填满窗体或页面，图片可能会变形）和 fmPictureSizeModeZoom（放大图片，但图片在水平和垂

直方向上都不变形）三种方式。

12.2.9 Flash 控件属性

Flash 控件是 ShockwaveFlash 控件的简称，它有 39 个属性，在实际工作中常用的属性仅两个，分别是“Movie”和“EmbedMovie”属性。

“Movie”属性代表 swf 格式的 flash 动画路径，例如“D:\动画\Logo.swf”。

“EmbedMovie”属性代表是否将动画嵌入工作簿。如果赋值为 True，那么表示将动画文件嵌入工作簿，其优点是在发送文件给其他用户时只需要发送单个文件，使用更方便，缺点是文件体积会变大；如果赋值为 False，那么窗体只是通过“Movie”属性引用指定路径下的动画文件，工作簿与动画文件间是相互独立的，在将文件发送给其他用户时需要将两个文件一起发送出去，如果路径不对则可能无法播放。

如果 Flash 动画体积较小，那么最好将 Flash 文件嵌入工作簿。

12.2.10 批量设置控件的属性

批量设置控件的属性分两种情况，一是添加控件后对多个同类控件设置相同属性，二是在添加控件前制作一个控件模板，预设其所有属性，以后需要将此控件时插入控件模板即可。

1．添加控件后

当有多个同类控件需要设置相同属性时，可以在窗体中拖曳鼠标光标从而选择多个控件，然后在属性窗口中设置它们的共有属性。例如在图 12-60 中有 3 个选项按钮，拖曳鼠标光标选择 3 个选项按钮后再将它们的“Caption”属性设置为“合格”，将“Left”属性设置为 5，将“Width”属性设置为 40，最终效果如图 12-61 所示。

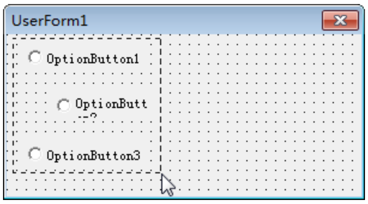


图 12–60 多选控件

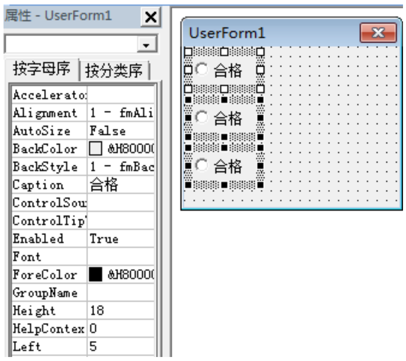


图 12–61 将三个控件设置相同的文字、左边距和宽度

2．添加控件前

在添加控件前，可以先设计一个控件模板，以后直接使用模板可节约操作时间。

假设需要设置一个用于关闭窗体的“取消”按钮模板，操作步骤如下。

（1）单选择菜单“插入”→“用户窗体”，然后在窗体中添加一个命令按钮。

（2）在属性窗口将命令按钮设置好高度、宽度、需要显示的文字，并且将“Cancel”属性设置为 True。

（3）将设置好的命令按钮拖曳至工具箱，如图 12-62 所示。

（4）当下次需要创建相同的命令按钮时，在工具箱中选择控件模板，然后拖曳到窗体中即可，

如图 12-63 所示。此方式创建的命令按钮将得到与图 12-62 完全一样的外观和文字，以及相同的“Cancel”属性。

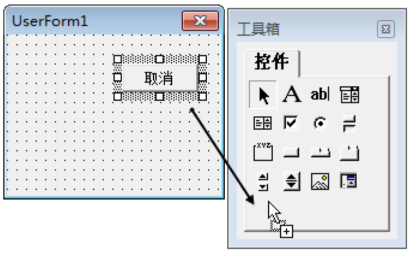


图 12–62　将控件模板拖到工具箱

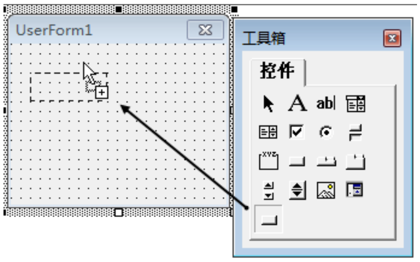


图 12–63　将控件模板从工具箱拖曳窗体

12.3　窗体与控件的事件

窗体中的 UserForm 对象和所有控件都拥有多个事件，脱离事件后，UserForm 对象和所有控件都没有存在价值。本节重点介绍 UserForm 对象和常用控件的常用事件。

12.3.1　UserForm 对象的事件

UserForm 对象的事件是指 UserForm 对象在满足内部设定的条件时所触发的事件，例如在关闭窗体前触发 UserForm 对象的 QueryClose 事件，激活窗体后触发 UserForm 对象的 Activate 事件。

UserForm 对象有 22 个事件，事件名称与触发条件如表 12-2 所示。

表 12-2　UserForm对象的事件名称与触发条件

事件名称	触发条件（在何时执行这个事件）
AddControl	在将控件插入到窗体时
Activate	激活窗体时
BeforeDragOver	在窗体中执行拖曳操作时
BeforeDropOrPaste	即将在一个对象上放置或粘贴数据时
Click	在窗体中用鼠标单击控件时
DblClick	在窗体中双击鼠标时
Deactivate	在窗体失去焦点时
Error	在控件检测到一个错误，并且不能将该错误信息返回调用程序时
Initialize	加载窗体之后、显示这个窗体之前
KeyDown	在按下任意键时
KeyUp	在某键弹起时
KeyPress	在用户按下ANSI键时（即按钮后能产生某字符时）
Layout	在修改窗体的位置时
MouseDown	在按下任意鼠标键时
MouseUp	释放鼠标按键时
MouseMove	在移动鼠标光标时
QueryClose	在关闭窗体之前
RemoveControl	从窗体中删除一个控件时
Resize	在更改窗体的大小时
Scroll	在按下滚动条时
Terminate	在关闭窗体之后
Zoom	在修改窗体的Zoom 属性时（即修改窗体的缩放比例时）

在以上 22 个事件中，常用的事件是 Activate 事件，它通常用于激活窗体时对窗体中的某些控件的属性赋值，例如指定复合框、列表框的宽度与边距、设置复选框的默认值，以及设置 Flash 动画的路径等。

12.3.2　控件的事件

控件的种类很多，每个控件拥有的事件也有很多，不过其中大部分事件与 UserForm 对象的事件相同，所以本节不再一一罗列每类控件的所有事件，而是有针对性地介绍其中几个常用事件的功能与用法，并提供相关案例。

可以通过以下步骤获知每类控件所支持的所有事件的名称，并且从帮助中找到这些事件的含义与触发条件。

（1）在窗体中插入多类控件，例如复合框、命令按钮、列表框和文本框。

（2）双击窗体进入窗体代码窗口中，在窗口上方将看到两个复合框。单击左方的复合框将会弹出如图 12-64 所示的对象列表，其中包含了 UserForm 对象和窗体中的所有控件名称。

（3）从对象列表中选择“CheckBox1”，然后单击右侧的“过程”复合框，将弹出如图 12-65 所示的事件过程名称列表，此列表中包含了复合框控件所支持的所有事件名称。

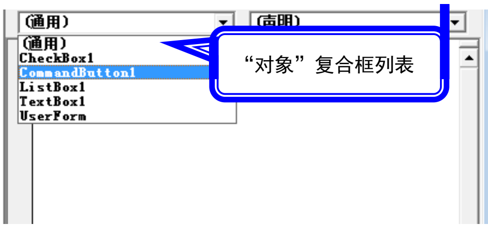


图 12–64　对象列表



图 12–65　事件名称列表

（4）通过步骤（3）了解到控件有哪些事件后，可以在帮助中搜索每个事件名称，从而了解事件的含义和触发条件。例如需要了解 Change 事件，那么按下【F1】键打开帮助窗口，在搜索框中输入“Change 事件”并按【Enter】键即可看到关于该事件的解释。

所有控件的事件加起来有几十个，其中常用事件包括：Change、DblClick、Enter、Exit、MouseMove、Click 等，接下来逐一讲解 Change、DblClick、Enter、Exit 和 MouseMove 等事件。由于 Click 事件比较简单，而且前面已经出现过多次，此处不再单独讲述。

1．Change 事件

Change 事件是复选框、选项按钮、文本框、复合框、列表框、滚动条、切换按钮等控件的共有事件，当修改控件的“Value”属性时触发此事件。

Change 事件通常用于在输入字符时限制字符类型，并且能每输入一个字符就校验一次。以下案例即为文本框的 Change 事件的应用。

【案例要求】在窗体中输入姓名和工号，在录入姓名的文本框中严禁输入数字和标点符号，在输入工号的文本框中则只允许输入数字。

【知识要点】TextBox1_Change 事件。

【操作步骤】

（1）选择菜单“插入”→“用户窗体”，然后在窗体中添加两个标签控件、两个文本框和两个命令按钮，并按图 12-66 所示的方式排列。



图 12-66　窗体控件布局

（2）双击窗体进入窗体的代码窗口，删除自动产生的代码，然后录入以下新代码。

```
'修改第一个文本框的值时执行，此过程是 TextBox1 对象的 Change 事件过程
Private Sub TextBox1_Change()   '放置位置：窗体的代码窗口中
    If Len(TextBox1) > 0 Then     '如果文本框中有字符
  '如果文本框的右边一位是 0 到 9 的数字，以及“.?/\<>,();'@#$%^&!、。_+:""|{}[]-”之
  '类的标点符号
        If Right(TextBox1, 1) Like "[0-9.?/\<>,();'@#$%^&!、。_+:""|{}]" Or _
        Right(TextBox1, 1) = "[" Or Right(TextBox1, 1) = "]" Or Right(TextBox1, 1) = "-" Then
            MsgBox "请不要输入数字和标点。", vbOKOnly, "友情提示"    '提示用户
            TextBox1.Value = Left(TextBox1, Len(TextBox1) - 1)          '删除右边一位
        End If
    End If
End Sub
```

此代码表示在修改第一个文本框的值时，检查文本框右边的字符是否为 0 到 9 的数字，或是“.?/\<>,();'@#$%^&!、。_+:""|{}[]-”之类的标点符号。如果是则提示用户，并且删除该字符，等待用户重新输入新值。

```
'修改第二个文本框的值时执行，此过程是 TextBox2 对象的 Change 事件过程
Private Sub TextBox2_Change()    '放置位置：窗体的代码窗口中
    If Len(TextBox2) > 0 Then      '如果文本框中有字符
        '如果文本框的右边一位不是 0 到 9 的数字
        If Right(TextBox2, 1) Like "[!0-9]" Then
            MsgBox "请不要输入数字以外的字符", vbOKOnly, "友情提示"      '提示用户
            TextBox2.Value = Left(TextBox2, Len(TextBox2) - 1)              '删除右边一位
        End If
    End If
End Sub
```

此代码表示在修改第二个文本框的值时，检查文本框右边的字符是否是 0 到 9 以外的字符，如果是 0 到 9 以外的字符则提示用户，然后删除该字符，等待用户重新输入新值。

```
'单击第一个命令按钮时执行，此过程是 CommandButton1 对象的 Click 事件过程
Private Sub CommandButton1_Click()'放置位置：窗体的代码窗口中
    MsgBox "代码从略", vbOKOnly, "友情提示"
End Sub
'单击第二个命令按钮时执行，此过程是 CommandButton2 对象的 Click 事件过程
Private Sub CommandButton2_Click()
    Unload Me
End Sub
```

（3）在代码窗口中按下【F5】键运行窗体，在第一个文本框中录入“张 3”，由于字符 3 不符合既定的规则，所以会弹出如图 12-67 所示的提示信息。

（4）在第二个文本框中录入“058 九”，由于前三个字符符合规则，第四个字符不符合规则，所以将弹出如图 12-68 所示的提示信息。

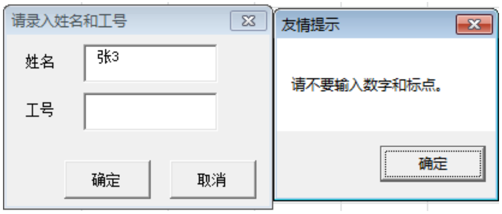


图 12–67　禁止输入数值或者标点

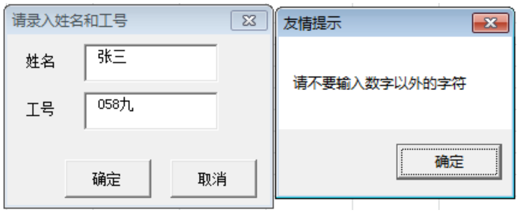


图 12–68　禁止输入数值以外的字符

【代码分析】

（1）在单元格中输入数据时可以通过数据有效性设置来实现字符类型校验，但不可能每输入一个字符校验一次。改用窗体中录入字符，然后将录入的值导出到指定单元格即可有效地防错，限制录入指定类型以外的字符。

（2）Like 运算符的比较对象中使用“[]”时表示范围，用符号“-”来标识起止边界。例如“[0-9]”表示从 0 到 9 的十个字符，“[a-zA-Z]”表示二十六个小写字母和大写字母。

由于“[”、“]”和“-”三个字符都有着特殊的含义，所以不能在“[]”之间限制这三个字符本身。在本例代码中加入了 Or 运算符，将“[”、“]”和“-”三个字符按三个条件分别处理。

（3）文本框的 Change 事件是在录入字符之后触发的，它与 Enter 事件、KeyDown 事件和 KeyPress 事件既有相同点也有不同点。

Enter 事件：控件接收到焦点之前触发事件，例如当前焦点在文本框 A 中，单击文本框 B 或者在文本框 A 中按下【Enter】键进入文本框 B，此时将触发文本框 B 的 Enter 事件。

KeyDown 事件：在按下任意键时触发事件。

KeyPress 事件：在用户按下一个【ANSI】键时触发事件，包括输入任何可打印的键盘字符、按下【Ctrl】键与标准字母表中字符的组合键、【Ctrl】键与任何特殊字符的组合键、【Backspace】键和【Esc】键。如果在文本框中按下【Delete】键或者【Enter】键、【Tab】键，以及在中文输入法状态下输入汉字等不会触发 KeyPress 事件。

本例案例文件请参考：..\ 第 12 章\ 12-17　文本框的 Change 事件应用.xlsm

2．DblClick 事件

DblClick 事件表示双击时触发的事件。可用于在双击图像控件时重新设置图片路径，或者在双击列表框控件时更新列表框的引用范围等。

以下事件是列表框的 DblClick 事件应用。

【案例要求】激活窗体时在列表框中显示工作表的所有数据，允许用户双击列表框重新指定其数据源。

【知识要点】UserForm_Activate 和 ListBox1_DblClick 事件。

【操作步骤】

（1）选择菜单“插入”→“用户窗体”，然后在窗体中添加一个列表框控件。

（2）双击窗体进入窗体的代码窗口，删除自动产生的代码，然后录入以下新代码。

```
'显示窗体时执行，此过程是 UserForm 对象的 Activate 事件过程
Private Sub UserForm_Activate()                        '放置位置：窗体的代码窗口中
    Dim Rng As Range                                   '声明 Range 型变量
    If Not IsEmpty(ActiveSheet.UsedRange) Then         '如果工作表不是空表
        Set Rng = ActiveSheet.UsedRange                '将已用区域赋值给变量 Rng
```

```
            ListBox1.Width = Rng.Columns.Count * 40 + 10        '设置列表框的宽度，等于每列列宽 40 加 10
            ListBox1.ColumnCount = Rng.Columns.Count            '指定列表框显示的列数
            Me.Width = ListBox1.Width + 20                      '指定窗体的宽度为列表框的宽度加 20
            '为列表框的每一列指定宽度为 40
            ListBox1.ColumnWidths = Replace(WorksheetFunction.Rept("40,",
    Rng.Columns.Count) & ",", ",,", "")
            ListBox1.RowSource = Rng.Address                    '指定列表框的数据源
        End If
    End Sub
```

此代码表示在激活窗体时将工作表的所有数据显示在列表框中，列表框的列数、宽度以及窗体的宽度随工作表中数据的列数而变化。

```
    '双击列表框时执行，此过程是 ListBox1 对象的 DblClick 事件过程
    Private Sub ListBox1_DblClick(ByVal Cancel As MSForms.ReturnBoolean)
        Dim Rng As Range                                        '声明 Range 型的变量
        On Error Resume Next                                    '当程序出错时继续执行下一步
        Set Rng = Application.InputBox("请选择区域", "区域", , , , , , 8)  '让用户重新指定列表框的数据源
        If Err <> 0 Then Exit Sub                   '如果有错误则退出程序（表示用户单击了“取消”按钮）
       '获取 Rng 的第一个区域与已用区域的交集（列表框的数据源只支持单区域）
        Set Rng =Intersect(Rng.Areas(1), ActiveSheet.UsedRange)
        '如果不存在交集则提示用户，然后结束过程
        If Rng Is Nothing Then MsgBox "请选择有数据的区域", vbOKOnly, "友情提示": Exit
    Sub
        ListBox1.Width = Rng.Columns.Count * 40 + 10            '设置列表框的宽度为列数乘以 40 加 10
        ListBox1.ColumnCount = Rng.Columns.Count                '设置列表框的列数
        Me.Width = ListBox1.Width + 20                          '设置窗体的宽度为列表框的宽度加 20
         '为列表框的每一列指定宽度为 40
        ListBox1.ColumnWidths = Replace(WorksheetFunction.Rept("40,", Rng.Columns.Count) & ",", ",,", "")
        ListBox1.RowSource = Rng.Address                        '指定列表框的数据源
    End Sub
```

此代码表示在双击列表框时让用户重新指定列表框的数据源，同时根据用户选择的区域大小调整列表框和窗体的宽度。

（3）在代码窗口中按下【F5】键运行窗体，在窗体中将罗列出工作表中的所有数据，如图 12-69 所示。

（4）双击列表框任意位置，在弹出“区域”对话框后选择 A1:C8 区域，对话框中将出现所选区域的地址，如图 12-70 所示。

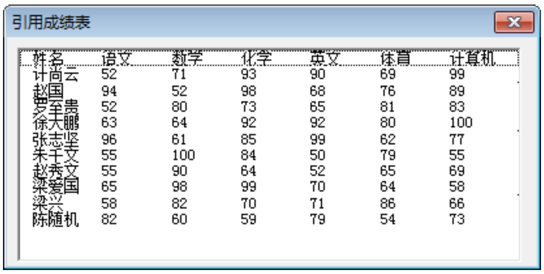


图 12–69　激活窗体时在列表框显示所有数据

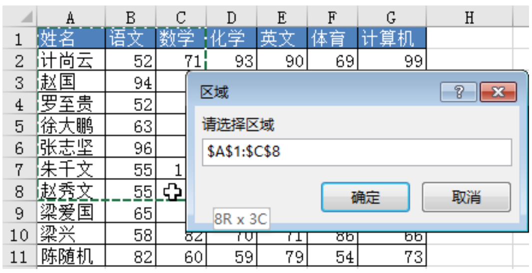


图 12–70　双击列表框重置数据源

在“区域”框中单击“确定”按钮后，列表框将显示为如图 12-71 所示效果。

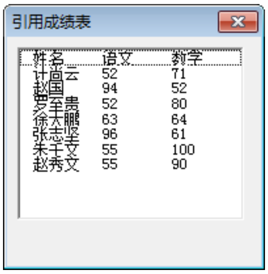


图 12–71　更新数据源后的列表框效果

【代码分析】

（1）双击列表框时触发 ListBox1_DblClick 事件，但使用鼠标右键双击不能触发此事件。

（2）列表框的“RowSource”属性只支持文本，所以不能赋值为“Rng”，而是“Rng.Address”。

（3）列表框的“RowSource”属性不支持多区域，所以在使用 Intersect 方法合并 Rng 与已用区域时采用“Intersect(Rng.Areas(1), ActiveSheet.UsedRange)”，表示不管用户选择多少个区域，只取第 1 个区域。

本例案例文件请参考：..\ 第 12 章\ 12-18 列表框的 DblClick 事件应用.xlsm

3. Enter 与 Exit 事件

Enter 事件与 Exit 事件分别在获得焦点与失去焦点时触发，善用这两个事件可以让程序更自动化。以下案例是文本框的 Enter 事件与 Exit 事件的应用。

【案例要求】在窗体中分别输入日期、机器编号和机器异常现象 3 项内容，输入任意数据后只需要按【Enter】键即可输入下一项，全程不需要使用鼠标。另外，要求每输入一项内容都有相关的内容提示。

【知识要点】TextBox1_Enter 和 TextBox3_Exit 事件。

【操作步骤】

（1）选择菜单“插入”→“用户窗体”，然后在窗体中添加 4 个标签和 3 个列表框控件，布局如图 12-72 所示。

（2）双击窗体，进入窗体的代码窗口，删除自动产生的代码，然后录入以下新代码。

```
'文本框获得焦点时执行，此过程是 TextBox1 对象的 Enter 事件过程
Private Sub TextBox1_Enter()   '放置位置：窗体的代码窗口中
    '第 1 个文本框获得焦点时，让标签显示当前内容相关的提示信息
    Label4.Caption = "提示：" & Chr(10) & "    请输入日期，格式必须是“YYYY-MM-DD”"
End Sub
'文本框获得焦点时执行，此过程是 TextBox2 对象的 Enter 事件过程
Private Sub TextBox2_Enter()'放置位置：窗体的代码窗口中
    '第 1 个文本框获得焦点时，让标签显示当前内容相关的提示信息
    Label4.Caption = "提示：" & Chr(10) & "    机器编号只能是两位数值，以“#”结束，不可输入文本"
End Sub
'文本框获得焦点时执行，此过程是 TextBox3 对象的 Enter 事件过程
Private Sub TextBox3_Enter()'放置位置：窗体的代码窗口中
    '第 1 个文本框获得焦点时，让标签显示当前内容相关的提示信息
    Label4.Caption = "提示：" & Chr(10) & "    请将机器故障现象录入完整"
End Sub
```

以上 3 个过程表示在文本框获得焦点时，修改标签 Label4 的“Caption”属性，从而使其显

示与当前录入对象相关的主题说明。

```
'文本框失去焦点时执行，此过程是 TextBox3 对象的 Exit 事件过程
Private Sub TextBox3_Exit(ByVal Cancel As MSForms.ReturnBoolean)
    '如果 3 个文本框都是内容
    If Len(TextBox1.Value) > 0 And Len(TextBox2.Value) > 0 And
Len(TextBox3.Value) > 0 Then
        With Cells(Rows.Count, 1).End(xlUp)        '引用 A 列最后一个非空单元格
            .Offset(1, 0) = TextBox1.Value          '对下方一个单元格赋值为第 1 个文本框的值
            '对下移一个单位、右移一个单位的单元格赋值为第 2 个文本框的值
            .Offset(1, 1) = TextBox2.Value
            '对下移一个单位、右移两个单位的单元格赋值为第 3 个文本框的值
            .Offset(1, 2) = TextBox3.Value
        End With
    End If
    TextBox1 = ""                                   '清除第 1 个文本框
    TextBox2 = ""                                   '清除第 2 个文本框
    TextBox3 = ""                                   '清除第 3 个文本框
End Sub
```

以上过程表示在文本框 TextBox3 失去焦点时，将 3 个文本框的值导入单元格。

（3）在代码窗口中按下【F5】键运行窗体，默认第 1 个文本框有焦点，所以右方的标签控件显示日期相关的提示内容，如图 12-73 所示。

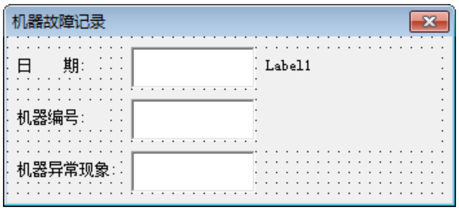


图 12–72　窗体控件布局

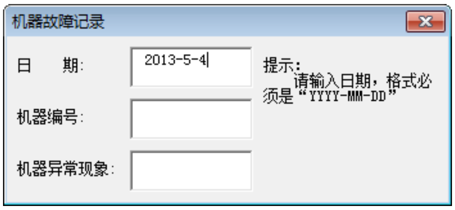


图 12–73　显示日期提示

（4）当输入日期后按下【Enter】键，鼠标光标定位到第 2 个文本框中，此时右方的标签控件将显示与机器编号相关的提示信息。当输入机器编号并单击【Enter】键后，标签控件马上更新为与机器异常现象相关的提示信息，效果分别如图 12-74 和图 12-75 所示。

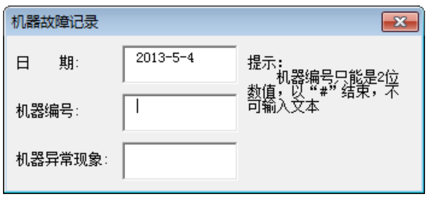


图 12–74　显示机器编号提示

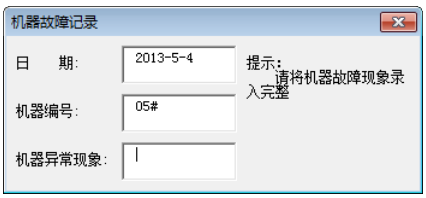


图 12–75　显示机器异常现象提示

（5）在第 3 个文本框中输入数据并单击【Enter】键，程序会将 3 个文本框的数据导入工作表，然后自动清空 3 个文本框的值，并且鼠标光标定位到第 1 个文本框，等待输入第 2 笔资料。

【代码分析】

（1）文本框的 Enter 事件表示在文本框获得焦点时触发的事件。在第 1 个文本框中按下【Enter】键后，鼠标光标将定位到第 2 个文本框中，此时第 1 个文本框触发 Exit 事件，第 2 个文本框触发 Enter 事件。

（2）本例中在第 3 个文本框中按下【Enter】键后，鼠标光标重新定位到第 1 个文本框，此

时可触发 TextBox3 的 Exit 事件。通过 TextBox3_Exit 事件将文本框的值导入工作表可使窗体更简洁，避免使用命令按钮，同时也减少了单击按钮的时间，从而提高输入效率。

本例案例文件请参考：..\ 第 12 章\ 12-21 文本框与命令按钮的 Enter、Exit 事件.xlsm

4．MouseMove 事件

MouseMove 事件是指鼠标指针从控件上方移过时触发的事件，类似于网页中鼠标指针经过带有超链接的网址时网址会变色及显示下划线。所以本节模仿网页的思路设计一个鼠标指针移过标签时将标签控件的文字显示为加粗、倾斜状态且同步显示网址的案例应用。

【案例要求】制作一个《Excel 2013 VBA 编程与实践》和《Excel 2016 实用技巧自学宝典》两本书的介绍窗体，包括内容介绍和京东网、当当网、亚马逊网的订购地址。当鼠标指针指向“京东网”标签时，要突出该标签，同时显示网址；若单击标签则可以打开网址。

【知识要点】Label2_MouseMove、Label2_Click 和 Frame1_MouseMove 事件。

【操作步骤】

（1）选择菜单“插入”→“用户窗体”，将“Caption”属性修改为“新书推荐”，然后在窗体中添加一个多页控件。

（2）将多页控件的第 1 页和第 2 页名称分别修改为“Excel 2013 VBA 编程与实践”和“Excel 2016 实用技巧自学宝典”。

（3）在第 1 页中添加一个标签，在其“Caption”属性中写上相关图书的内容介绍；

（4）在第 1 页中添加一个框架，将框架的“Caption”属性赋值为“订购方式”，然后在框架中添加 3 个标签，代表 3 个网址。

（5）在多页控件的下方再添加一个标签用于显示提示信息，标签的“Caption”属性为“友情提示：鼠标指针指向定购方式时将显示网址，单击可打开网址”，如图 12-76 所示。

在属性窗口中输入“Caption”属性值时不能换行，所以在窗体中选择标签，然后将鼠标光标定位到冒号之后，按下【Ctrl+Enter】组合键实现换行。

（6）将第 1 页中的第 1 个标签控件复制到第 2 页中，然后修改其中的图书内容介绍。

此时窗体与控件的布局方式如图 12-77 所示。

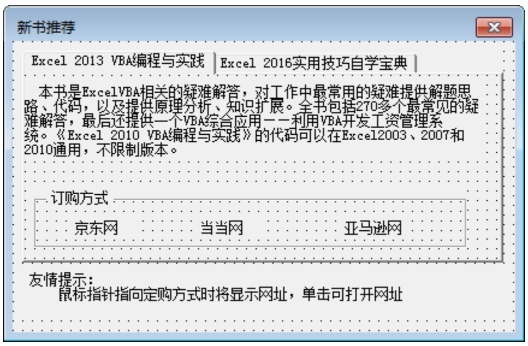


图 12–76　窗体第 1 页的控件布局方式

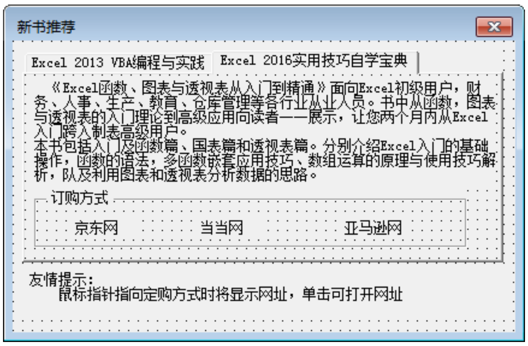


图 12–77　窗体第 2 页的控件布局方式

（7）选择菜单“插入”→“模块”，在模块中录入以下代码。

```
'根据代表图书的参数和代表链接的参数决定打开哪一个网址
'其中第一参数的可选值是 1 和 2，代表第 1 本书和第 2 本书
'第二参数的可选值是 10、20、30，分别代表第 1 个网站（京东网）、第 2 个网站（当当网）和第 3
```

```
个网站（亚马逊网）
Sub 链接(Book As Byte, WebAddress As Byte)
     Select Case Book + WebAddress
    Case 11   '如果代表图书的编号加上代表网址的编号等于 11，那么打开以下网址
              Shell "explorer.exe ""http://item.jd.com/11739787.html"""
    Case 21   '如果代表图书的编号加上代表网址的编号等于 21，那么打开以下网址
        Shell "explorer.exe ""http://product.dangdang.com/23744435.html"""
    Case 31   '如果代表图书的编号加上代表网址的编号等于 31，那么打开以下网址
        Shell "explorer.exe ""https://www.amazon.cn/dp/B012DRHTNS/ref=sr
_1_6?ie=UTF8&qid=1484008412&sr=8-6"""
     Case 12   '如果代表图书的编号加上代表网址的编号等于 12，那么打开以下网址
        Shell "explorer.exe ""http://item.jd.com/11914778.html"""
    Case 22   '如果代表图书的编号加上代表网址的编号等于 22，那么打开以下网址
        Shell "explorer.exe ""http://product.dangdang.com/23949541.html"""
    Case 32   '如果代表图书的编号加上代表网址的编号等于 32，那么打开以下网址
        Shell "explorer.exe ""https://www.amazon.cn/dp/B01EIE3L7C/ref=sr_
1_3?ie=UTF8&qid=1484008765&sr=8-3"""
    End Select
End Sub
```

（8）双击窗体进入窗体的代码窗口，删除自动产生的代码，然后录入以下代码。

```
Private Sub Label2_Click()          '单击 Label2 时执行
    Call 链接(1, 10)                '执行过程“链接”，其参数为 1 和 10，代表第 1 本书的第 1 个网址
End Sub
'鼠标指针在标签 Label2 上移过时执行
Private Sub Label2_MouseMove(ByVal Button As Integer, ByVal Shift As Integer,
ByVal X As Single, ByVal Y As Single)
    Label2.Font.Bold = True         '让 Label2 的字体加粗显示
    Label2.Font.Italic = True       '让 Label2 的字体倾斜显示
    '修改标签 Label5 的显示文字为网页地址
    Label5.Caption = "http://item.jd.com/11739787.html"
End Sub
Private Sub Label3_Click()          '单击 Label3 时执行
    Call 链接(1, 20)                '执行过程“链接”，其参数为 1 和 20，代表第 1 本书的第 2 个网址
End Sub
'鼠标指针在标签 Label3 上移过时执行
Private Sub Label3_MouseMove(ByVal Button As Integer, ByVal Shift As Integer,
ByVal X As Single, ByVal Y As Single)
    Label3.Font.Bold = True         '让 Label3 的字体加粗显示
    Label3.Font.Italic = True       '让 Label3 的字体倾斜显示
     '修改标签 Label5 的显示文字为网页地址
    Label5.Caption = "http://product.dangdang.com/23744435.html"
End Sub
Private Sub Label4_Click()          '单击 Label4 时执行
    Call 链接(1, 30)                '执行过程“链接”，其参数为 1 和 30，代表第 1 本书的第 3 个网址
End Sub
'鼠标指针在标签 Label4 上移过时执行
Private Sub Label4_MouseMove(ByVal Button As Integer, ByVal Shift As Integer,
ByVal X As Single, ByVal Y As Single)
    Label4.Font.Bold = True         '让 Label4 的字体加粗显示
    Label4.Font.Italic = True       '让 Label4 的字体倾斜显示
    '修改标签 Label5 的显示文字为网页地址
    Label5.Caption = "https://www.amazon.cn/dp/B012DRHTNS/ref=sr_1_6?ie
```

```
=UTF8&qid=1484008412&sr=8-6"
End Sub
'鼠标指针在 Frame1 上移过时执行
Private Sub Frame1_MouseMove(ByVal Button As Integer, ByVal Shift As Integer,
ByVal X As Single, ByVal Y As Single)
    Label2.Font.Bold = False      'Label2 的字体不再加粗显示
    Label2.Font.Italic = False    'Label2 的字体不再倾斜显示
    Label3.Font.Bold = False      'Label3 的字体不再加粗显示
    Label3.Font.Italic = False    'Label3 的字体不再倾斜显示
    Label4.Font.Bold = False      'Label4 的字体不再加粗显示
    Label4.Font.Italic = False    'Label4 的字体不再倾斜显示
    '修改标签 Label5 的显示文字
    Label5.Caption = "友情提示:" & Chr(10) & "      鼠标指针指向定购方式时将显示网址，
单击可打开网址"
End Sub
Private Sub Label7_Click()        '单击 Label7 时执行
    Call 链接(2, 10)              '执行过程“链接”，其参数为 2 和 10，代表第 2 本书的第 1 个网址
End Sub
'鼠标指针在标签 Label7 上移过时执行
Private Sub Label7_MouseMove(ByVal Button As Integer, ByVal Shift As Integer,
ByVal X As Single, ByVal Y As Single)
    Label7.Font.Bold = True       '让 Label7 的字体加粗显示
    Label7.Font.Italic = True     '让 Label7 的字体倾斜显示
    '修改标签 Label5 的显示文字为网页地址
    Label5.Caption = "http://item.jd.com/11914778.html"
End Sub
Private Sub Label8_Click()        '单击 Label8 时执行
    Call 链接(2, 20)              '执行过程“链接”，其参数为 2 和 20，代表第 2 本书的第 2 个网址
End Sub
'鼠标指针在标签 Label8 上移过时执行
Private Sub Label8_MouseMove(ByVal Button As Integer, ByVal Shift As Integer,
ByVal X As Single, ByVal Y As Single)
    Label8.Font.Bold = True       '让 Label8 的字体加粗显示
    Label8.Font.Italic = True     '让 Label8 的字体倾斜显示
    '修改标签 Label5 的显示文字为网页地址
    Label5.Caption = "http://product.dangdang.com/23949541.html"
End Sub
Private Sub Label9_Click()        '单击 Label9 时执行
    Call 链接(2, 30)              '执行过程“链接”，其参数为 2 和 30，代表第二本书的第三个网址
End Sub
'鼠标指针在标签 Label9 上移过时执行
Private Sub Label9_MouseMove(ByVal Button As Integer, ByVal Shift As Integer,
ByVal X As Single, ByVal Y As Single)
    Label9.Font.Bold = True       '让 Label9 的字体加粗显示
    Label9.Font.Italic = True     '让 Label9 的字体倾斜显示
    '修改标签 Label5 的显示文字为网页地址
    Label5.Caption = "https://www.amazon.cn/dp/B01EIE3L7C/ref=sr_1_3?ie=
UTF8&qid=1484008765&sr=8-3"
End Sub
'鼠标指针在 Frame2 上移过时执行
Private Sub Frame2_MouseMove(ByVal Button As Integer, ByVal Shift As Integer,
ByVal X As Single, ByVal Y As Single)
```

```
        Label7.Font.Bold = False        'Label7 的字体不再加粗显示
        Label7.Font.Italic = False      'Label7 的字体不再倾斜显示
        Label8.Font.Bold = False        'Label8 的字体不再加粗显示
        Label8.Font.Italic = False      'Label8 的字体不再倾斜显示
        Label9.Font.Bold = False        'Label9 的字体不再加粗显示
        Label9.Font.Italic = False      'Label9 的字体不再倾斜显示
        '修改标签 Label5 的显示文字
        Label5.Caption = "友情提示:" & Chr(10) & "      鼠标指针指向定购方式时将显示网址，单击可打开网址"
    End Sub
```

在以上 14 个过程中包括 6 个标签的 Click 事件和 MouseMove 事件，以及 2 个框架的 MouseMove 事件，分别实现在鼠标指向标签时将标签的文字加粗、倾斜显示，同时在底部显示网址；在鼠标指针离开标签时，标签的字体恢复正常状态。

（9）在代码窗口中按下【F5】键运行窗体，在窗体中默认会显示第 2 页（在窗体中插入多页控件后最后一次编辑的页面）。

（10）单击多页控件的第一页，在鼠标指针指向标签“京东网”时，标签将显示为加粗、倾斜状态，同时在对话框底部显示第 1 本书的第 1 个网址，效果如图 12-78 所示；

（11）鼠标指针离开第 1 个标签，将鼠标指针移至空白处，此时“京东网”标签将恢复正常状态，同时窗体底部的标签不再显示网址；

（12）在鼠标指针指向“当当网”时，“当当网”标签将显示为加粗、倾斜状态，同时在对话框底部显示第 1 本书的当当网购买网址，效果如图 12-79 所示。

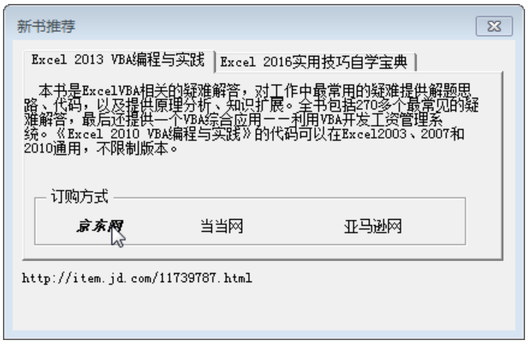


图 12-78　鼠标指针指向第 1 本书的京东网购买网址

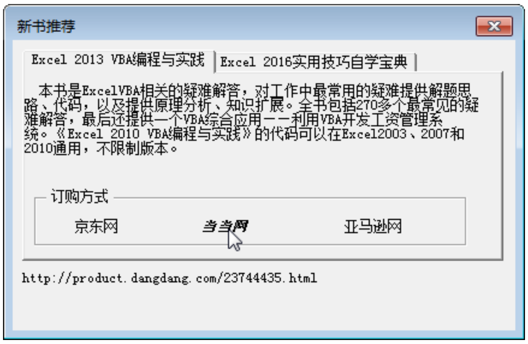


图 12-79　鼠标指针指向第 1 本书的当当网购买网址

【代码分析】

（1）当鼠标指针移到框架控件中的标签上时，将触发标签的 MouseMove 事件，当鼠标指针离开标签时则触发框架控件的 MouseMove 事件。

本例在标签控件的 MouseMove 事件中设置了标签的字符为加粗、倾斜，但是当鼠标指针离开标签后它并不会自动还原，所以本例将标签控件与框架控件的事件配合使用，当鼠标指针离开标签时，通过框架控件的 MouseMove 事件还原标签的字符显示状态。

（2）本例的思路相当简单，但是代码较长，因此显得复杂，读者在练习时可以仅使用 1 到 2 个标签，而不是 6 个。

（3）本例为了简化窗体中的代码，在模块中添加了一个过程“链接”，该过程根据参数决定打开哪一个网址。事实上也可以删除此过程，直接在每个标签的 Click 事件中用 Shell 函数打开网址。

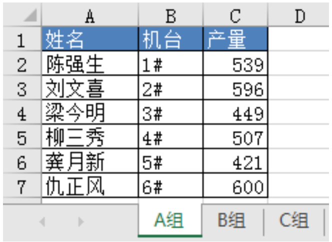
本例案例文件请参考：..\ 第 12 章\ 12-20 架框与文本框的 MouseMove 事件.xlsm

12.4 窗体应用实战

窗体在工作中的应用较多，制作登录界面、查询数据、快速录入面板、展示软件帮助信息、制作软件的参数选项对话框等方面都会用到窗体。本节展示数据查询、快速录入面板和制作软件的参数选项对话框三个方面案例。

12.4.1 开发多工作表查询窗体

【案例要求】图 12-80 包含多个工作表，每个工作表存放一个组别的生产数据，要求设计一个查询窗口，在窗口中录入查询条件后可以将所有工作表中符合条件的值在窗体中罗列出来，并可以将查询结果导入工作表的指定区域。



	A	B	C	D
1	姓名	机台	产量	
2	陈强生	1#	539	
3	刘文喜	2#	596	
4	梁今明	3#	449	
5	柳三秀	4#	507	
6	龚月新	5#	421	
7	仇正风	6#	600	

A组 B组 C组

图 12–80 产量表

【知识要点】标签、文本框、命令按钮、列表框、Click 事件、设置焦点 SetFocus、设置列表框的每列宽度 ColumnWidths 和返回列表框的列表 ListBox1.List。

【操作步骤】

（1）选择菜单“插入”→“用户窗体”，并将其“Caption”属性修改为“按产量条件查询职工信息”。

（2）在窗体中添加两个标签控件，两个命令按钮，一个文本框和一个列表框，其中命令按钮的“Caption”属性分别为“确定”和“导出”，前者用于执行查询，后者用于导入查询结果。

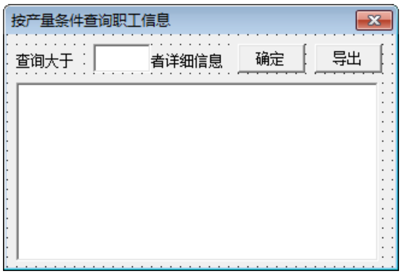


图 12–81 窗体控件布局

（3）双击窗体进入窗体的代码窗口，删除自动产生的代码，然后录入以下代码。

```
'单击命令按钮时执行，此过程是 CommandButton1 控件的 Click 事件过程
Private Sub CommandButton1_Click()          '放置位置：窗体的代码窗口中
    '声明 2 个变量，分别用于计算 3 个组别的最小产量和最大产量
     Dim MinValue As Integer, MaxValue As Integer
    '利用工作表函数 Min 计算最小产量
```

```
        MinValue = WorksheetFunction.Min(Sheets("A 组").Range("C:C"), Sheets("B
    组").Range("C:C"), Sheets("C 组").Range("C:C"))
        '利用工作表函数 Max 计算最大产量
        MaxValue = WorksheetFunction.Max(Sheets("A 组").Range("C:C"), Sheets("B
    组").Range("C:C"), Sheets("C 组").Range("C:C"))
        If Len(TextBox1.Value) = 0 Then                 '如果文本框是空白的
            MsgBox "请输入查询标准,数值在：" & MinValue & "到" & MaxValue & "之间",
    vbOKOnly, "友情提示"                                  '提示用户
        ElseIf Not IsNumeric(TextBox1.Value) Then       '如果文本框中的字符不是数值
            MsgBox "请输入" & MinValue & "到" & MaxValue & "之间的数值", vbOKOnly, "
    友情提示"
    '如果用户输入的值不在最大值与最小值之间
     ElseIf TextBox1.Value < MinValue Or TextBox1.Value >= MaxValue Then
            MsgBox "请输入" & MinValue & "到" & MaxValue & "之间的数值", vbOKOnly, "
    友情提示"
        Else                                            '否则
            Dim Arr(), Item                             '声明 2 个变量，1 个数组 1 个计数器
            Item = 1                                    '先将计数器设置为默认值等于 1
            ReDim Arr(1 To 4, 1 To Item)                '重置数组变量的维数和上标与下标
            Arr(1, 1) = "组别"                          '预设列表框的 4 个标题
            Arr(2, 1) = "姓名"
            Arr(3, 1) = "机台"
            Arr(4, 1) = "产量"
            Dim sht As Worksheet, Arr2, RowItem As Integer          '声明变量
            For Each sht In Worksheets                              '遍历所有工作表
              '将 A:C 区域与已用区域的交集的值赋予变量 Arr2
                Arr2 = Intersect(sht.Range("A:C"), sht.UsedRange)
                For RowItem = 2 To UBound(Arr2)                     '遍历数组的每一行（标题行除外）
                    '如果产量大于文本框中的值（文本框的值是文本，需要通过 Cint 函数转换成数值）
                    If Arr2(RowItem, 3) > CInt(TextBox1.Value) Then
                        Item = Item + 1                             '累加计数器
                        ReDim Preserve Arr(1 To 4, 1 To Item)       '重新指定最末维的上标
                        Arr(1, Item) = sht.Name                     '向数组中写入组别
                        Arr(2, Item) = Arr2(RowItem, 1)             '向数组中写入姓名
                        Arr(3, Item) = Arr2(RowItem, 2)             '向数组中写入机台号
                        Arr(4, Item) = Arr2(RowItem, 3)             '向数组中写入产量
                    End If
                Next RowItem
            Next sht
            If Item > 1 Then                                        '如果计数器大于 1
                Me.ListBox1.ColumnWidths = "50,60,50,50"            '分别指定列表框的 4 列列宽
                ListBox1.List = WorksheetFunction.Transpose(Arr)    '将数组写入列表框
            Else                                                    '否则
                MsgBox "没找到此范围的产量信息", vbOKOnly, "友情提示"
            End If
    End If
    CommandButton2.SetFocus                                         '将焦点转移到第二个命令按钮
End Sub
```

以上过程表示在单击“确定”按钮时在工作表中执行跨表查询，以文本框中录入的值为查询条件。如果有符合条件的值，那么将它们逐一写入到数组变量 Arr 中，最后将数组变量 Arr 赋予

列表框的 List 属性。

为了方便用户，在查询完后将焦点转移到第 2 个命令按钮，可以减少一次按键。

```
'单击命令按钮时执行，此过程是 CommandButton1 控件的 Click 事件过程
Private Sub CommandButton2_Click()
    If Me.ListBox1.ListCount > 1 Then '如果列表框不是空白的
    Dim Rng As Range                       '声明一个 Range 型的变量
    On Error Resume Next                   '在程序执行出错时继续执行下一步
  '弹出一个对话框让用户选择区域
    Set Rng = Application.InputBox("请选择存放区域，单个单元格即可", "指定区域", , , , , , 8)
    If err <> 0 Then Exit Sub '如果有错误，那么退出程序（表示用户单击了“取消”按钮）
  '将 Rng 所代表的区域的左上角单元格重置为与列表框相同的宽度和高度，从而得到一个新的区域
  '然后将列表框的列表赋值给这个区域
    Rng(1).Resize(UBound(ListBox1.List)+1, UBound(ListBox1.List, 2)+1) = ListBox1.List
    End If
End Sub
```

以上过程表示在单击“导出”按钮时弹出一个输入框，供用户指定数据的保存地址。当指定区域后，根据列表框的行数和列数重置区域大小，并将列表框的所有值导出到该区域。

（4）在窗体的代码窗口中按下【F5】键运行窗体，在查询条件文本框中录入 500，表示查找大于 500 的产量，在按下【Enter】键后，焦点将转移到“确定”按钮。

（5）再次单击【Enter】键，在窗体中将罗列出在所有工作表中大于 500 的产量信息，同时焦点转移到“导出”按钮，效果如图 12-82 所示。

（6）再次单击【Enter】键，将弹出一个输入框，用于指定查询结果的保存区域。假设选择 E1 单元格，当单击“确定”按钮后，将列表框中的值显示在以 E1 开始的区域，效果如图 12-83 所示。

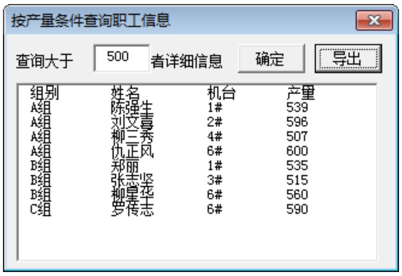
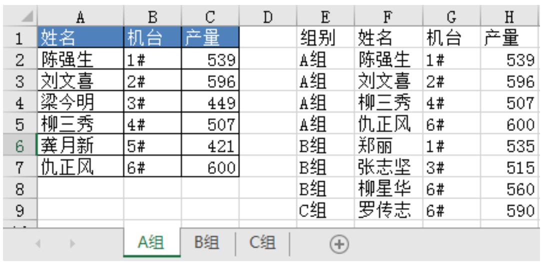


图 12–82　查询大于 500 的产量



	A	B	C	D	E	F	G	H
1	姓名	机台	产量		组别	姓名	机台	产量
2	陈强生	1#	539		A组	陈强生	1#	539
3	刘文喜	2#	596		A组	刘文喜	2#	596
4	梁今明	3#	449		A组	柳三秀	4#	507
5	柳三秀	4#	507		A组	仇正风	6#	600
6	龚月新	5#	421		B组	郑丽	1#	535
7	仇正风	6#	600		B组	张志坚	3#	515
8					B组	柳星华	6#	560
9					C组	罗传志	6#	590

图 12–83　导出查询结果

【代码分析】

（1）在执行查询前，有必要判断用户指定的查询标准值是否处于最高产量与最低产量之间，如果超过范围则直接结束程序，不必执行多余的比较过程。

（2）查询每个表的数据时都从第 2 行开始，跳过了标题行，所以在执行查询前需要将标题写入数组，否则在列表框中不会显示标题。同时，由于在声明数组变量时指定了数组的维数和上标、下标，所以此数组变量只能一个个写入值，不能使用 Array 函数产生一个一维数组，然后将数组赋值给数组变量。

（3）在文本框中输入的任意字符都是 String 型，所以本例的代码中使用了 CInt 函数将它转换为 Integer 型，否则不管输入任何值都会大于工作表中的产量值，从而无法查找到目标数据。

（4）在一个文本框中按下【Enter】键时会将焦点转移到另一个文本框或者它后面的命令按

钮上，但是当焦点在命令按钮上时，按下【Enter】键后却不会转移焦点。本例为了提升操作的速度，在“确定”按钮的事件过程中使用了代码“CommandButton2.SetFocus”，从而将焦点转移到“导出”按钮。

（5）“ListBox1.List”表示列表框 ListBox1 的所有数据，它是一个二维数组，所以可以利用 UBound 函数计算列表框的行数与列数。

（6）由于“ListBox1.List”的第一维和第二维下标都是 0，所以需要加 1，否则利用 Resize 对 Rng(1)重置区域后会出现少一行、少一列的情况 。

（7）在“CommandButton2_Click”过程中，如果用户指定区域时选择了多区域，那么在使用 Resize 对 Rng 重置区域大小时会出错。本例的解决办法是“Rng(1)”，即不管用户选择多少个的区域，只读取其中第 1 个单元格。

本例案例文件请参考：..\ 第 12 章\ 12-21 开发多工作表查询窗体.xlsm

12.4.2　开发多工作表快速录入面板

【案例要求】要求通过窗体向图 12-84 中的 3 个工作表中录入姓名、机台和产量信息，全程使用键盘输入，不需要使用鼠标，从而提升录入速度。在图 12-85 中有所有职工的姓名、机台和组别，允许调用此数据。



图 12–84　以“组别”命名的工作表



图 12–85　职员信息表

【知识要点】标签、文本框、命令按钮、Click 事件和设置焦点 SetFocus。

【操作步骤】

（1）选择菜单“插入”→“用户窗体”，并将其“Caption”属性修改为“快速录入面板”。

（2）在窗体中添加两个标签、两个文本框和两个命令按钮，并按图 12-86 所示的方式排放控件。



图 12–86　窗体控件布局方式

其中“取消”按钮的“Cancel”属性需设置为 True，表示可以按下【Esc】键关闭窗体。

（3）双击窗体进入窗体的代码窗口，删除自动产生的代码，然后录入以下代码。

```
'单击“确定”按钮时执行，此过程是 CommandButton1 按钮的 Click 事件过程
Private Sub CommandButton1_Click()  '放置位置：窗体的代码窗口中
```

```
        If Len(TextBox1.Value) = 0 Or Len(TextBox2.Value) = 0 Then    '如果未录入姓名和产量
            MsgBox "请将姓名和产量录入完整", vbOKOnly, "友情提示"     '提示用户
      '如果产量文本框中录入的值不是数值
        ElseIf Not IsNumeric(TextBox2.Value) Then
            MsgBox "产量只能是数值，请重新录入", vbOKOnly, "友情提示" '提示用户
            TextBox2 = ""                                             '清空第二个文本框
            TextBox2.SetFocus                                         '将焦点设置为第二个文本框
        Else
            Dim Arr, RowItem As Integer, Bl As Boolean, sht As Worksheet '声明变量
          '将“职员信息表”中的已用区域赋值给数组 Arr
            Arr = Worksheets("职员信息表").UsedRange.Value
            For RowItem = 2 To UBound(Arr)                            '遍历数组的第一行（标题除外）
              '如果录入的姓名与数组中第 RowItem 行第 1 列的值相同
                If TextBox1.Value = Arr(RowItem, 1) Then
                  '将与数组中第 RowItem 行第 3 列同名的工作表赋值给变量 Sht
                    Set sht = Worksheets(Arr(RowItem, 3))
                  '向 Sht 表中写入姓名
                    sht.Cells(Rows.Count, 1).End(xlUp).Offset(1, 0) = TextBox1.Value
                  '向 Sht 表中写入机台号
                    sht.Cells(Rows.Count, 1).End(xlUp).Offset(0, 1) = Arr(RowItem, 2)
                  '向 Sht 表中写入产量
                    sht.Cells(Rows.Count, 1).End(xlUp).Offset(0, 2) = TextBox2
                  '对 A1 的当前区域设置边框
                    sht.Cells(1, 1).CurrentRegion.Borders.LineStyle = xlContinuous
                    Bl = True                                         '将变量 Bl 的值设置为 True
                    Exit For
                End If
            Next RowItem
            If Bl = False Then                                        '如果 Bl 的值为 true
                MsgBox "资料库中不存在您录入的姓名，请重新录入", vbonly, "友情提示" '提示用户
            Else                                                      '否则
                sht.Activate                                          '激活工作表
            End If
            TextBox1 = ""                                             '清空第一个文本框
            TextBox2 = ""                                             '清空第二个文本框
            TextBox1.SetFocus                                         '让第一个文本框获得焦点
        End If
    End Sub
```

以上过程表示在单击“确定”按钮时将三个文本框中的字符导入对应的工作表。在导入前需要判断文本框中是否有字符，输入的产量值是否包含文本，以及输入的姓名是否有错别字等。

当在窗体中录入的姓名和产量导出到工作表中后，清空两个文本框，等待录入下一笔资料。

```
    '单击“CommandButton2”按钮时执行，此过程是 CommandButton2 按钮的 Click 事件过程
    Private Sub CommandButton2_Click()    '放置位置：窗体的代码窗口中
        Unload Me                         '关闭窗体
    End Sub
```

以上过程表示在单击“取消”按钮时关闭窗体。由于按钮 CommandButton2 的“Cancel”属性已设置为 True，所以也可以按下【Esc】键关闭窗体。

（4）在窗体的代码窗口中按下【F5】键运行窗体，在窗体中对姓名和产量分别输入“垄月新”和“500”，单击“确定”按钮后将弹出如图 12-87 所示提示，表示代码已检测到姓名录入错误。

（5）重新录入姓名“龚月新”和产量“500”，按两次【Enter】键后录入的姓名和产量将自动导入到目标工作表中，同时会自动产生机台号，自动清空窗体中的两个文本框，等待录入下一笔资料，效果如图 12-88 所示。

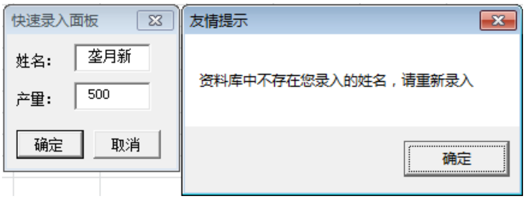


图 12–87　在录入错别字时弹出提示框

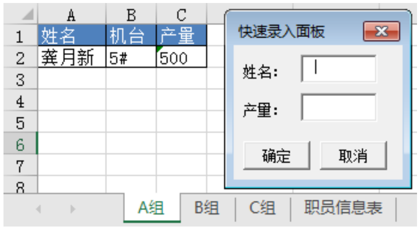


图 12–88　录入第一笔资料

（6）继续录入姓名“谢有金”和产量“720”，程序会自动激活与谢有金所在组别同名的工作表，然后将资料写入到该工作表中，同时清空窗体中的两个文本框。

【代码分析】

（1）由于产量只能是数值，所以在向工作表中写入数据前有必要利用 IsNumeric 函数判断用户录入的数据是否为数值，如果不是则提示用户重新录入。

（2）通过循环语句判断数组 Arr 的第一列中是否包含当前录入的姓名时，只要找到一个相等的值就可以终止循环，没有必要继续判断下去，所以在“End If”前需要加上“Exit For”。

（3）在文本框中按下【Enter】键后会自动转移焦点，但是在命令按钮中按【Enter】键却不会，所以有必要在命令按钮的 Click 事件中加上代码“TextBox1.SetFocus”。

本例案例文件请参考：..\ 第 12 章\ 12-22 开发多工作表快速录入面板.xlsm

12.4.3　以指定名称批量新建或复制工作表

【案例要求】批量新建或者复制工作表，可以自动调节工作表数量，并且选择新表插入到活动表之前还是之后，还可以选择新表名称来源于单元格中的数据还是“Sheet”加序号。

【知识要点】框架、标签、滚动条、命令按钮、选项按钮、Click 事件和 Change 事件。

【操作步骤】

（1）选择菜单“插入”→“用户窗体”，并将其“Caption”属性修改为“新建/复制工作表”。

（2）在窗体中插入两个标签、一个滚动条、六个选项按钮，分别存放在三个框架控件中，并且要注意顺序。在第一个框架中的选项按钮是 OptionButton1 和 OptionButton2，在第二个框架中的选项按钮是 OptionButton3 和 OptionButton4，第三个框架中的选项按钮是 OptionButton5 和 OptionButton6。最后再添加两个命令按钮，分别将“Caption”属性修改为“确定”与“取消”。所有控件排放方式如图 12-89 所示。

（3）将滚动条的“Min”和“Max”属性分别设置为 1 和 254。将“取消”命令按钮的“Cancel”属性设置为 True，然后将窗体的宽度调小，从而隐藏“取消”按钮，效果如图 12-90 所示。

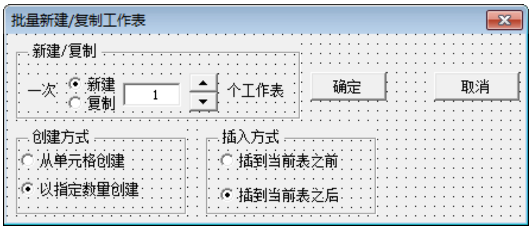


图 12–89　窗体控件布局方式

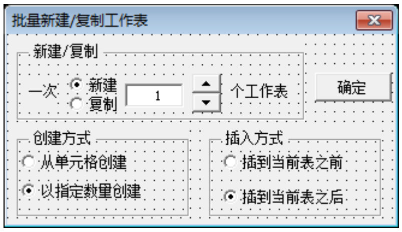


图 12–90　隐藏取消按钮

（4）双击窗体进入窗体的代码窗口，删除自动产生的代码，然后录入以下代码。

```
'单击 OptionButton3 选项按钮时执行，此过程是 OptionButton3 选项按钮的 Click 事件过程
Private Sub OptionButton3_Click()        '放置位置：窗体的代码窗口中
    TextBox1.Visible = False             '隐藏文本框 TextBox1
    SpinButton1.Visible = False          '隐藏滚动条 SpinButton1
    Label2.Visible = False               '隐藏标签 Label2
End Sub
'单击 OptionButton4 选项按钮时执行，此过程是 OptionButton4 选项按钮的 Click 事件过程
Private Sub OptionButton4_Click()        '放置位置：窗体的代码窗口中
    TextBox1.Visible = True              '显示文本框 TextBox1
    SpinButton1.Visible = True           '显示滚动条 SpinButton1
    Label2.Visible = True                '显示标签 Label2
End Sub
```

以上两个过程表示选择“从单元格创建”选项时隐藏文本框、滚动条和标签，因为在此选项下创建的工作表数量由单元格的数量决定，在选择“以指定数量创建”选项时可以将它们显示出来。

```
'单击 SpinButton1 滚动条时执行，此过程是 SpinButton1 滚动条的 Change 事件过程
Private Sub SpinButton1_Change()         '放置位置：窗体的代码窗口中
    TextBox1 = SpinButton1               '将滚动条的值赋予文本框
End Sub
```

此过程表示在单击滚动条时将滚动条的值赋予文本框，将此值作为创建工作表的数量。

```
'单击 CommandButton1 命令按钮时执行
'此过程是 CommandButton1 命令按钮的 Click 事件过程
Private Sub CommandButton1_Click()       '放置位置：窗体的代码窗口中
     '隐藏窗体界面，避免显示 Application.InputBox 所创建的输入框时被窗体挡住视线
    Me.Hide
    On Error Resume Next                 '如果有错误继续执行下一步
    Dim SheetIndex As Integer, ShtName As String, Rng As Range, Cell As Range
    '声明变量
    ShtName = ActiveSheet.Name           '记录活动工作表的名称，便于执行完毕后返回此工作表中
   '如果录入的值超过 255 则提示用户，然后退出程序（要注意“Exit Sub”与 If 语句在同一行）
    If TextBox1.Value > 255 Then MsgBox "一次插入工作表数不可大于 255 个。", 64, "
友情提示": Exit Sub
    If TextBox1.Value <= 0 Then MsgBox "一次插入工作表数不可小于等于 0 个。", 64, "
友情提示": Exit Sub
    If Len(TextBox1.Value) = 0 Then MsgBox "文本框不能空白", 64, "友情提示": Exit Sub
    If OptionButton1 Then                           '如果选择“新建”选项按钮
        If OptionButton4 Then                       '如果选择“以指定数量创建”选项按钮
            Application.ScreenUpdating = False      '关闭屏幕更新，加快程序执行效率
            SheetIndex = ActiveSheet.Index          '记录活动工作表的序号
            If OptionButton5 Then                   '如果选择“插入到工作表之前”选项按钮
```

```
                '新建工作表，插入到活动工作表之前，数量等于文本框中的值
                Worksheets.Add Before:=Worksheets(SheetIndex), Count:=
TextBox1.Value
            Else '否则
                '新建工作表，插入到活动工作表之后，数量等于文本框中的值
                 Worksheets.Add After:=Worksheets(SheetIndex), Count:=
TextBox1.Value
            End If
            Application.ScreenUpdating = True           '恢复屏幕刷新
        Else                                            '否则（选择“以单元格创建”）
           Set Rng = Application.InputBox("请选择区域", "工作表名称来源", , 10, 10, , , 8)
            SheetIndex = ActiveSheet.Index              '记录活动工作表的序号
            For Each Cell In Rng                        '遍历 Rng 中的所有单元格
                '如果 cell 是空单元格，那么提示用户，然后退出程序
                If Len(Cell) = 0 Then MsgBox "不能选择空白单元格！", 64, "友情提
示": Exit Sub
                Application.ScreenUpdating = False      '关闭屏幕刷新，加快程序执行效率
                If OptionButton5 Then                   '如果选择“插入到工作表之前”选项按钮
               '创建一个新表，插入到活动表之前
                With Worksheets.Add(Before:=Worksheets(ShtName))
                    SheetIndex = SheetIndex + 1         '重新指定变量所代表的序号
                     .Name = Cell.Text                  '用单元格 Cell 的值对新表重新命名
                  End With
                Else                                    '否则（表示选择“插入到工作表之后”选项按钮）
'创建一个新表，插入到活动表之后
                    With Worksheets.Add(After:=Worksheets(SheetIndex))
                        SheetIndex = SheetIndex + 1     '重新指定变量所代表的序号
                        .Name = Cell.Text               '用单元格 Cell 的值对新表重新命名
                    End With
                End If
            Next
        End If
        Sheets(ShtName).Select                          '选择执行代码前的活动工作表
        Unload Me                                       '关闭窗体
        Application.ScreenUpdating = True               '恢复屏幕更新
    Else                                                '否则（选择“复制”）
        Dim j As Integer                                '声明一个 Integer 型的变量
        If OptionButton4 Then                           '如果选择“以指定数量复制”选项按钮
            Application.ScreenUpdating = False          '关闭屏幕更新，加快程序执行效率
            SheetIndex = ActiveSheet.Index - 1          '记录活动工作表的序号
            For j = 1 To TextBox1.Value                 '启动循环，从 1 到文本框所指定的值
                If OptionButton5 Then                   '如果选择“插入到工作表之前”选项按钮
                    '复制变量 ShtName 所代表的工作表到活动表之前
                    Sheets(ShtName).Copy Before:=Worksheets(SheetIndex + 1)
                    SheetIndex = SheetIndex + 1         '重新指定变量所代表的序号
                Else                                    '否则（表示选择“插入到工作表之后”选项按钮）
                    '复制变量 ShtName 所代表的工作表到第 SheetIndex + 1 个工作表之后
                    Sheets(ShtName).Copy After:=Worksheets(SheetIndex + 1)
                End If
            Next j
            Application.ScreenUpdating = Tru            '恢复屏幕更新
```

```
            Else                                              '否则(选择“以单元格复制”)
    '弹出对话框，让用户选择区域
                Set Rng = Application.InputBox("请选择区域", "工作表名", , , , , , 8)
                Application.ScreenUpdating = False            '关闭屏幕更新，加快程序执行效率
                SheetIndex = ActiveSheet.Index                '记录活动工作表的序号
                For Each Cell In Rng                          '遍历 Rng 区域中所有单元格
                    '如果单元格是空白的，那么提示用户，然后跳转至 err 标签处
                    If Len(Cell) = 0 Then MsgBox "不能选择空白单元格！", 64, "友情提示": GoTo err
                    If OptionButton6 Then                     '如果选择“插入到工作表之前”选项按钮
                        '复制变量 ShtName 所代表的工作表到第 SheetIndex 个工作表之前
                        Sheets(ShtName).Copy Before:=Sheets(SheetIndex)
                        SheetIndex = SheetIndex + 1           '重新指定变量所代表的序号
                        ActiveSheet.Name = Cell.Text          '用 Cell 单元格的值对活动工作表命名
                    Else                                      '否则(选择“插入到工作表之后”选项按钮)
                        '将变量 ShtName 所代表的工作表复制到第 SheetIndex 个工作表之后
                        With Sheets(ShtName).Copy(After:=Sheets(SheetIndex))
                            SheetIndex = SheetIndex + 1 '重新指定变量所代表的序号
                            ActiveSheet.Name = Cell.Text '用 Cell 单元格的值对活动工作表命名
                        End With
                    End If
                Next
            End If
            Worksheets(ShtName).Select                        '选择执行代码前的活动工作表
    err:                                                      '设置一个标签
            Unload Me                                         '卸载窗体
            Application.ScreenUpdating = True                 '恢复屏幕更新
        End If
    End Sub
```

以上过程表示在单击“确定”按钮时根据用户的选择，决定是新建还是复制工作表，以及决定工作表数量、创建方式和插入方式。

```
    '单击 CommandButton2 命令按钮时执行
    '此过程是 CommandButton2 命令按钮的 Click 事件过程
    Private Sub CommandButton2_Click()
     Unload Me    '卸载窗体
    End Sub
```

以上过程表示在单击“取消”按钮时关闭窗体。由于按钮已被隐藏起来，所以实际使用时可按【Esc】键调用此段代码。

（5）在窗体的代码窗口中按下【F5】键运行窗体，在文本框中录入 4，其他选项保持默认即可，表示一次性新建 4 个工作表，保存在活动工作表之后。图 12-91 为设置界面，图 12-92 为运行结果。

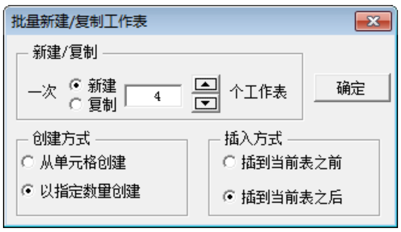


图 12–91　设置界面



图 12–92　运行结果

（6）删除 Sheet1 以外的所有工作表，然后在 Sheet1 的 A1:A7 区域分别录入“周一”到“周日”7 个字符串。

（7）再次运行窗体，将创建方式设置为“从单元格创建”，将插入方式设置为“插到当前表之前”，如图 12-93 所示。然后单击“确定”按钮，程序会弹出“工作表名称来源”的对话框，选择 A1:A7 区域后单击“确定”按钮，即可创建 7 个工作表，且引用 A1:A7 中的 7 个单元格的值作为工作表名称，运行结果如图 12-94 所示。

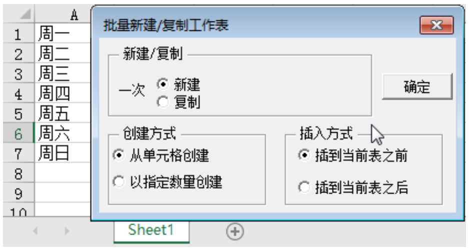


图 12-93　设置界面



图 12-94　运行结果

【代码分析】

（1）滚动条和文本框的默认属性都是“Value”，所以忽略属性时都表示“Value”属性。“TextBox1 = SpinButton1”其实相当于“TextBox1.Value = SpinButton1.Value”。

（2）在设计窗体时控件的插入顺序很重要。例如在本例中使用了 6 个选项按钮，每个框架中有 2 个选项按钮，它们的命名分别是 OptionButton1 到 OptionButton6。在编写代码时要根据选项按钮的命名来判断它所代表的功能，所以需要注意顺序。当然最好的办法是将按钮的“名称”属性赋值为汉字，例如将 OptionButton1 的“名称”修改为“新建”，将 OptionButton2 的“名称”修改为“复制”，这样在调用选项按钮时出错的机率将会大幅降低。

（3）由于插入方式总是以活动工作表为基准，所以不管用何种方式插入工作表，在插入前皆有必要记录活动工作表的序号，即 ActiveSheet.Index。

（4）每创建一个新表就会更新屏幕一次，在批量创建工作表时会影响代码的执行效率，所以在每个循环语句之前需加入“Application.ScreenUpdating = False”，待循环结束后再恢复屏幕更新。

（5）Worksheets.Add 方法和 Sheets.Add 方法的 Count 参数都不支持 1 到 255 以外的数值，所以在代码中需要判断用户录入的值是否在可接受范围之内。

本例案例文件请参考：..\ 第 13 章\ 13-25 指定名称批量新建或复制工作表.xlsm

12.5　课后思考

1．如何设置才能在文本框中输入多行字符？如何才能在标签控件中输入多行字符？

2．选项按钮具有排它性，即在选择一个选项按钮后其他选项按钮即自动调节为未选中状态，如何能实现在窗体中同时选中多个选项选钮？

3．简述 Enter 事件与 Click 事件的相同点与不同点。

4．设计一个存放工作表目录的窗体，在窗体启动时自动在列表框中显示所有工作表的目录，单击列表框中的工作表名称时可以激活对应的工作表。

5．设计一个窗体，窗体中包含一个文本框和一个“确定”按钮，在文本框中只能录入 5 位数值的学号。请用代码限制用户录入的值必须是数值，每输入一位校验一次，而且在按下【Enter】键时要判断录入的值是否为 5 位数的学号。

第 13 章　定义 Ribbon 功能区选项卡

微软公司从 Excel 2007 版本开始使用功能区选项卡替代传统的工具栏和菜单。

从功能区选项卡第一次出现到现在已经过了十多年。实践证明功能区选项卡远比传统菜单更实用，既美观又操作方便，而且可以同时在相同的空间中放置更多的按钮，节约查找命令按钮的时间。对于 VBA 开发者而言，定制属于自己的专用功能区选项卡可提升调用过程的便捷性，同时也使自己的程序可以更好地融入到 Excel 中去。

本章详细阐述功能区选项卡中各部件的开发思路，同时教学制作功能区选项卡模板，从而提升开发效率。

本章要点

- ◆ 功能区选项卡开发基础
- ◆ Ribbon 定制之语法分析
- ◆ 使用回调函数强化功能区选项卡
- ◆ 使用模板

13.1　功能区选项卡开发基础

开发功能区选项卡之前先做一些准备工作，包括了解功能区的特性与结构、定制功能区的方法和安装代码编辑器。

13.1.1　Ribbon 的特点

功能区选项卡的英文名称是 Ribbon，它的外形就像一条飘浮在工作表顶端的带子。

功能区选项卡将 Excel 的常用功能按用户需求分布在 9 个选项卡中，分别为“文件”、“开始”、“插入”、“页面布局”、“公式”、“数据”、“审阅”、“视图”和“开发工具”。其中“开始”选项卡包含最常用的功能命令，打开 Excel 后默认显示“开始”选项卡界面；“开发工具”选项卡属于高级用户专用，所以默认处于隐藏状态，需要在“Excel 选项”对话框中手动调整其显示状态。

功能区中的任何一个命令按钮都支持快捷键操作，例如【Alt+h+m+c】组合键表示调用“开始”选项卡中的“合并后居中”，【Alt+r+p+s】组合键表示调用“审阅”选项卡中的“保护工作表”……在利用代码创建新的选项卡和按钮后，也同样支持快捷键操作。

功能区选项卡虽然是 Excel 的功能之一，但是不能用 VBA 开发功能区选项卡和命令按钮。

13.1.2　功能区的组件图示

功能区包含选项卡、组、命令按钮、切换按钮、标签、复选框、文字框、弹出式菜单、拆分按钮、下拉列表控件、分隔线和对话框启动器等组件。不过这些组件不会同时显示在一个界面中，在图 13-1 中包含了功能区中的多数控件，读者可以从图 13-1 中了解各组件的外观。

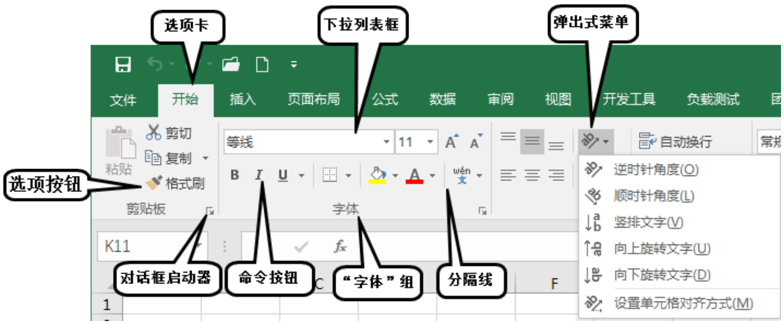


图 13–1　常见的功能区组件图示

13.1.3　手动定制功能区

从 Excel 2007 版本开始可以使用功能区，但从 Excel 2010 版本之后才支持手动定制功能区。

按下【Alt+t+o】组合键打开“Excel 选项”对话框，单击左侧的“自定义功能区”，即可看到手动定制功能区选项卡的界面。在此界面中可以新建选项卡、新建组，以及新建命令按钮。

不过手动定制的功能区远远不能满足需求，因为手动定制所产生的功能区组件并没有集成到文件中，所以当把 Excel 文件发送出去后，其他电脑打开文件无法看到定制的组件，仍然只能使用【Alt+F8】组合键调用过程。

所以，虽然 Excel 允许手动定制功能区，但实际工作中 VBA 开发者们都采用 XML 代码定制，而非手动定制。

13.1.4　认识 Ribbon 代码编辑器

功能区代码是 XML 语言，无法通过 VBA 代码创建功能区选项卡和选项卡中的各种组件。

本章主要介绍通过外置软件来定制功能区，软件全名叫作“Custom UI Editor for Microsoft Office 2010”，即 Excel 2010 专用的功能区界面编辑器（兼容后来的 Excel 2013 和 Excel 2016），简称为 Custom UI Editor。访问以下网址可以下载此软件：

```
http://pan.baidu.com/s/1gdrBlcf
```

当安装好软件后，将在开始菜单中显示“Custom UI Editor for Microsoft Office 2010”，单击即可打开软件。图 13-2 是 Custom UI Editor 软件的操作界面。

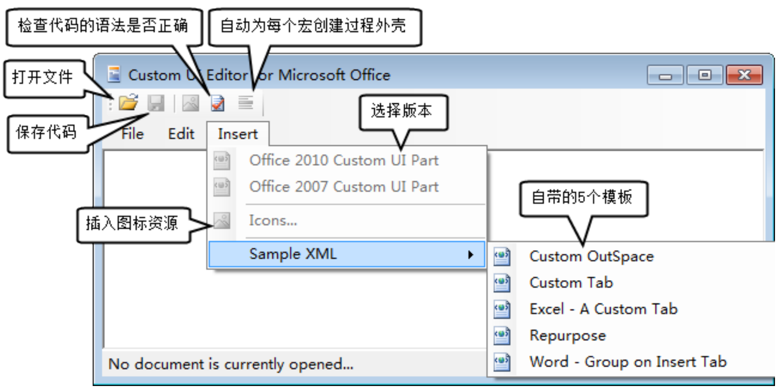


图 13–2　Custom UI Editor 软件界面

虽然软件是英文界面的，但是只有 5 个按钮，就算用户不认识英文也可以正常操作。

第一个按扭（图标: ）用于打开文件。打开一个 xlsx 或者 xlsm 格式的 Excel 文件，然后再写入定制功能区的代码，最后单击第二个按钮保存代码，这样才可以在 Excel 文件中创建功能区相关的组件。

第三个按钮（图标: ）用于插入图片，当需要对命令按钮分配自定义图标时使用。不过建议调用 Excel 的内部图标资源，一方面可以减小文件体积，另一方面有利于提高打开文件的速度。

第四个按钮（图标: ）用于检查用户输入的代码是否存在语法问题，如果有语法错误将提示错误原因以及位置。引号不配对或者名称中含有非法字符等都属于语法错误。

第五个按钮（图标: ）用于产生代码中每个回调过程的程序外壳。由于回调过程都有若干个参数，而这些参数名称很难记忆，因此软件提供此功能对开发者而言帮助较大。

菜单“Insert”中的前两个菜单分别代表生成 Excel 2010 格式的 XML 文件和 Excel 2007 格式的 XML 文件。如果在单击第一个菜单后输入定制功能区的代码，那么此文件只能在 Excel 2010 中正常显示自定义的功能区组件；如果单击第二个菜单后输入定制功能区的代码，那么此文件用 Excel 2007 和 Excel 2010 打开都会正常显示自定义的功能区组件。考虑到兼容性，编写代码应采用 Excel 2007 格式。

当打开一个 xlsx 或者一个 xlsm 格式的文件后，单击第二个子菜单“Office 2007 Custom UI Part”，将会在该文件中插入一个存放功能区代码的文件，名为“customUI.xml”，如图 13-3 所示；如果单击第一个子菜单“Office 2010 Custom UI Part”，将插入一个名为“customUI14.xml”的文件，如图 13-4 所示，此文件中的代码所创建的功能区不能在 Excel 2007 中正常显示。

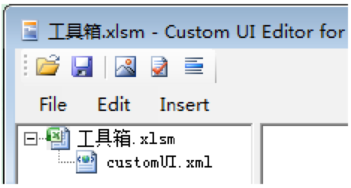


图 13–3 Excel 2007 格式的代码文件

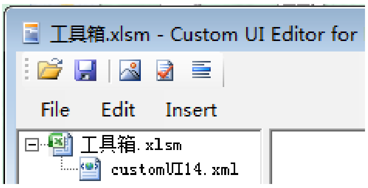


图 13–4 Excel 2010 格式的代码文件

Excel 2007 的功能区代码和更高版本的功能区代码的第一句并不相同，前者代码如下：

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
```

后者代码如下：

```
<customUI xmlns="http://schemas.microsoft.com/office/2009/07/customui">
```

Custom UI Editor 软件自带的 5 个模板中的代码是 Excel 2010 及更高版本专用的。

13.1.5 获取内置按钮图标

在开发功能区按钮时需要为按钮设置图标，既可以调用外置图标也可以调用内置图标。调用内置图标的效率最高，因此笔者本人一般不使用外置图标。

为了方便读者调用图标名称，笔者制作了一个 Excel 内置图标查看器，如图 13-5 所示。打开此工作簿后将自动生成一个新的选项卡“图标浏览”，在该选项卡中每页显示 20 个内置图标，在单击图标时会在活动单元格中产生该图标的名称。选择左侧的“前一页”和“下一页”，可以查看更多的图标。

本例案例文件请参考：..\ 第 13 章\ 13-1 内置图标浏览器.xlsm

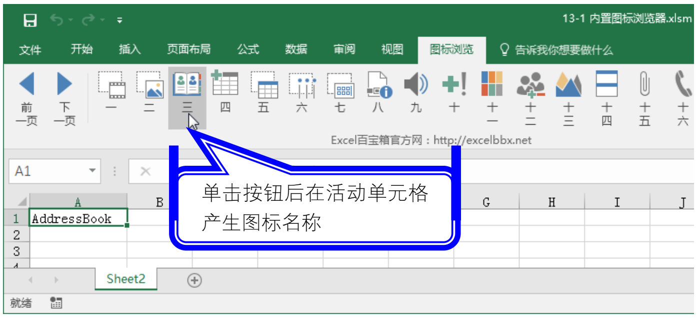


图 13–5　内置图标浏览器

13.2　Ribbon 定制之语法分析

功能区中的控件种类较多，修改不同类型的控件需要用不同的语法。

本节对功能区中的各类控件逐一分析语法，并提供效果图示。在学习本章前请安装 Custom UI Editor 软件。

13.2.1　功能区代码的结构

功能区代码的结构相当严谨，每句代码的顺序都有严格的规定，而且所有代码都必须配对，且严格区分大小写。

功能区代码的结构比较复杂，通过一段具体的代码来理解则相对容易一些，代码如下。

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
 <ribbon>
  <tabs>
   <tab id=  label=  >
    <group id=  label=  >
      <button id=  label=  screentip=  supertip=  onAction=  image=  />
      <menu id=  label=  screentip=  supertip=  size=   image=  >
        <button id=  label=  screentip=  supertip=  onAction=  image=  />
      </menu>
      <dialogBoxLauncher>
        <button id=  label=  screentip=  supertip=  onAction=  image=  />
     </dialogBoxLauncher>
    </group>
   </tab>
  </tabs>
 </ribbon>
</customUI>
```

以上代码用于在功能区中创建新的选项卡（等号右边未赋值），在新选项卡中创建弹出式菜单，在弹出式菜单中创建一个按钮。可以按以下方式理解这段代码的结构。

第一句和最后一句是配对的，它是功能区代码的根或者称之为壳，类似于 VBA 中 Sub 与 End Sub 的关系。首尾两句都包含“customUI”，表示这是一个自定义功能区的容器。首句代码使用了括号“<>”，末句代码使用配对的“</>”表示结束。缺少“<>”开头或者缺少“</>”结尾都会产生错误。其他任何表示容器的语句均遵循此规则。不过命令按钮、标签、分隔条等最底层的控件不属于容器，它们不需要遵循此原则。

代码中的“2006/01”是通用于 Excel 2007、Excel 2010、Excel 2013 和 Excel 2016 的，如

果需要编写 Excel 2010 及以上版本专用的功能区代码，应改用“2009/07”。

第二句与倒数第二句是配对的，“<ribbon>”代表后面的代码用于定制功能区，“</ribbon>”代表功能区定制过程结束。

第三句与倒数第三句是配对的，“<tabs>”表示它后面的代码作用于选项卡，“</tabs>”则表示定制选项卡的代码结束。

第四句与倒数第四句是配对的，“<tab>”表示此代码用于添加新选项卡或者修改内置选项卡，后面的 id 参数代表选项卡的 id，而 label 参数则表示选项卡显示在屏幕上的名称，可以随意自定义。在此模板中，等号后面故意留空是为了方便理解，表示此处可以根据实际需求赋值。模板中的 id 是指创建一个新的选项卡并为其指定一个 id，如果要引用内置的选项卡，那么需要将模板中的 id 修改为 idMso，例如“idMso="TabHome"”表示引用内部的“开始”选项卡。

第五句与倒数第五句是配对的，“<group>”表示此代码用于创建一个组，后面的 label 用于指定组的名称。当出现“</group>”时表示定制组的代码结束。

第六句代码“<button/>”用于创建一个命令按钮，由于按钮是最底层的元素，所以只需要一行代码，行首为“<”，行尾为“/>”。此处也可以添加其他的控件，包括命令按钮、切换按钮、标签、复选框、文字框、弹出式菜单、拆分按钮、下拉列表控件、分隔条等。

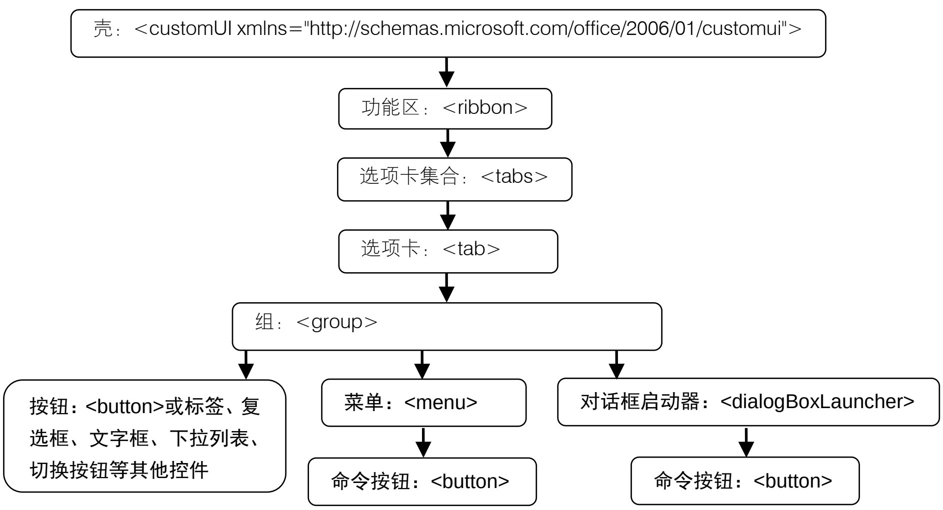
第七句与第九句是配对的，“<menu>”表示添加一个弹出式菜单，label、screentip、supertip、size、image 分别代表该弹出式菜单的菜单名称、提示信息、详细提示、大小和图标。当出现“</menu>”时表示定制弹出式菜单的代码结束。

第八句用于创建一个命令按钮，所以只需要一行代码。由于此按钮放置在“<menu>”与“</menu>”之间，表示它是弹出式菜单中的一个子菜单，而前面的“<button/>”放在“<group>”之后，表示它位于组中而不是弹出式菜单中。

第九句与倒数第六句又是配对的，“<dialogBoxLauncher>”代表创建一个对话框启动器。对话框启动器是组 group 的子元素，与弹出式菜单 menu 是同级别的对象。

第十句用于创建一个命令按钮，此按钮处于对话框启动器 dialogBoxLauncher 中。一个对话框启动器中只能放置一个命令按钮，但是一个组 group 或者一个弹出式菜单 menu 中可以放置多个命令按钮。

可以使用以下架构图来展示以上功能区部件的结构，从而有助于理解部件与部件之间的关系与顺序。



13.2.2　显示与隐藏功能区：ribbon

功能区的代码是 ribbon，控制功能区显示与隐藏的语法如下：

```
<ribbon startFromScratch="AA">
</ribbon>
```

其中“"AA"”属于占位符，为方便叙述而存在。此处若赋值为“true”则表示隐藏功能区，赋值为“false”或者直接忽略 startFromScratch 参数则表示显示功能区。

要注意定制功能区的代码是严格区分大小写的，“true”和“false”的每一个字母皆为小写。

实现隐藏功能区的具体操作如下（后续实现其他功能时仅讲解定制功能区的代码，不再详细描述操作步骤，可以参照此处的步骤操作，替换代码即可）。

（1）新建一个 Excel 文件，并将它保存为“隐藏功能区.xlsm”；

（2）打开 Custom UI Editor 软件，从 Custom UI Editor 软件中打开刚才保存的 Excel 文件。

（3）选择菜单“Insert”→“Office 2007 Custom UI Part”，从而插入一个 customUI.xml 文件；

（4）在右侧的代码窗口中录入以下代码（由于这几句代码很常用，所以可以将它保存在记事本中，需要用时再复制过来）。

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="true">
  </ribbon>
</customUI>
```

（5）单击“保存”按钮，然后关闭 Custom UI Editor 软件。

（6）双击打开刚才保存的 Excel 文件，此时的 Excel 功能区将处于隐藏状态。

图 13-6 是 Custom UI Editor 软件操作界面，图 13-7 是最终的隐藏功能区的效果。

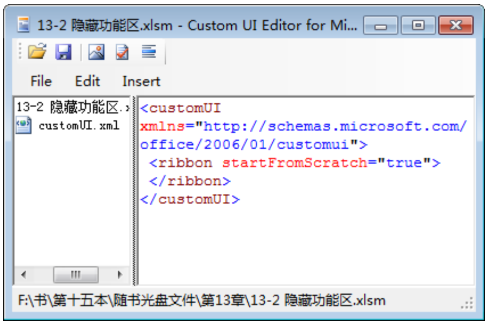


图 13–6　Custom UI Editor 软件操作界面

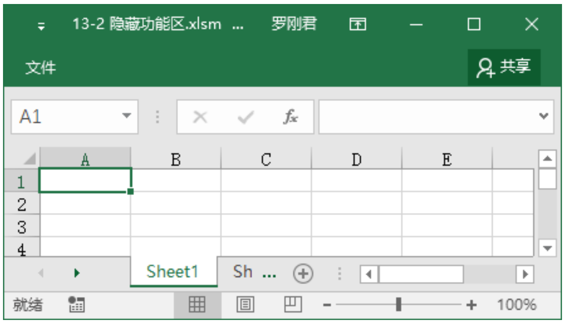


图 13–7　隐藏功能区的效果

注意：创建功能区代码的语言不是 VBA 语言，而是 XML 语句。在 xlsm 和 xlsx 格式的工作簿中都可以存放 XML 代码，从而对工作簿定制功能区，但是 xlsx 格式的工作簿不能保存 VBA 代码，而定制功能区的代码总是与 VBA 代码搭配应用才能体现其价值，所以在工作中请尽量将文件保存为 xlsm 格式。

本例案例文件请参考：..\ 第 13 章\ 13-2　隐藏功能区.xlsm

13.2.3　创建新选项卡：tab

【语法】

```
<tab id="AA" visible ="BB" label="CC" insertAfterMso="DD" insertBeforeMso= "EE"   keytip="FF" >
</tab>
```

其中参数 id 表示新选项卡的 id，必须为选项卡指定唯一的 id 名称，不能与其他选项卡同名；visible 代表选项卡的可见状态，在赋值为 False 时表示隐藏选项卡；label 表示选项卡显示在屏幕上的名称，允许重名；insertAfterMso 表示将新选项卡放置在此参数所指定的选项卡之后；insertBeforeMso 表示将新选项卡放置在此参数所指定的选项卡之前，不能与 insertAfterMso 同时出现，当同时忽略这两个参数时，表示将新选项卡放置在最后位置；keytip 表示选项卡的加速键，也称快捷键。

语法表中的 AA、BB、CC、DD、EE 等都是占位符，在编写代码时可根据需求进行修改。

在对 id、Label 和 keytip 参数赋值时，不区分大小写；在对其他参数赋值时，由于皆采用内部常量，所以必须区分大小写。

visible、label、insertAfterMso、insertBeforeMso 和 keytip 等为可选参数。

所有参数之间没有顺序要求，任何一个写在前面都可以。不过有三点值得注意：其一，参数与参数之间需要至少一个空格；其二，在对参数赋值时必须使用半角的引号；其三，逻辑值必须全部小写。

创建选项卡时通常要用到 insertAfterMso 参数，表示新选项卡放在指定内置的选项卡后方。Excel 的内置选项卡如表 13-1 所示，其中不包含上下文选项卡（即选中某个对象后才会出现的选项卡）。

表 13-1 内置选项卡名称

idMso名称	选项卡名称	idMso名称	选项卡名称
TabHome	开始	TabData	数据
TabInsert	插入	TabReview	审阅
TabPageLayoutExcel	页面布局	TabView	视图
TabFormulas	公式	TabDeveloper	开发工具

【案例】创建名为“Excel 精灵”的选项卡，显示在“开始”选项卡之后，加速键为 B。

【代码】

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
 <ribbon startFromScratch="false">
  <tabs>
   <tab id="NewTab" visible="true" label=" Excel 精灵" insertAfterMso ="TabHome" keytip="B">
   </tab>
  </tabs>
 </ribbon>
</customUI>
```

【效果】代码运行效果如图 13-8 所示。

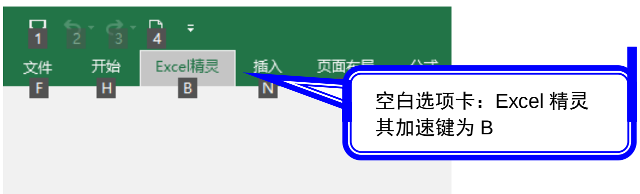


图 13–8 在【开始】选项卡之后创建新选项卡

调用加速键的方法是按【Alt】键。

本例案例文件请参考：..\ 第 13 章\ 13-3 创建新选项卡.xlsm

13.2.4　创建新组：group

【语法】

```
<group id="AA" visible="BB" label="CC" insertAfterMso="DD" insertBeforeMso ="EE">
</group>
```

语法表中 id、visible、label 参数与创建选项卡时的参数规则一致，不同的是 insertAfterMso 和 insertBeforeMso 两个参数。只有当新组位于内置选项卡中时，才需要使用这两个参数之一来表示新组的位置，否则直接忽略参数即可。

【案例】在“Excel 精灵”选项卡中创建一个名为“财务工具”的新组，在“开始”选项卡的“字体”组之前也创建一个为“财务工具”的新组。

【代码】

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
 <ribbon startFromScratch="false">
  <tabs>
   <tab id="NewTab" visible="true" label=" Excel 精灵" insertAfterMso="TabHome" keytip="B">
    <group id="Group1" visible="true" label="财务工具">
    </group>
   </tab>
   <tab idMso="TabHome" visible="true">
    <group id="Group2" visible="true" label="财务工具" insertBeforeMso ="GroupFont">
    </group>
   </tab>
  </tabs>
 </ribbon>
</customUI>
```

由于第一个新组处于自定义的选项卡中，所以不需要指定位置。第二个新组放在“开始”选项卡的“字体”组之后，所以指定 tab 的 id 时改用“idMso”，同时对 insertBeforeMso 参数赋值为“GroupFont”，即“字体”组。图 13-9 和图 13-10 分别是两个新组的外观（在 Excel 2016 中不会显示空白的组，因此本例采用 Excel 2010 界面截图，只为向读者展示代码创建的组的位置和效果）。

【效果】

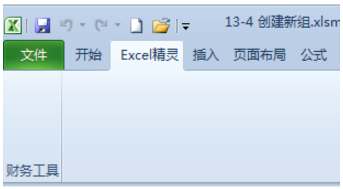


图 13–9　在“Excel 精灵”选项卡中添加“财务工具”组

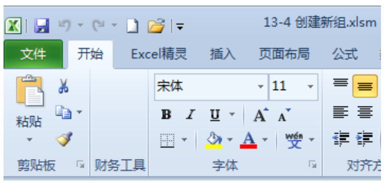


图 13–10　在“开始”选项卡插入新组

在表 13-2 中罗列了 Excel 中绝大部分的组，读者在编写代码时可以调用这些组名称。

表 13-2　内置组的名称

组　名	说　明	组　名	说　明
GroupClipboard	剪贴板	GroupCalculation	计算
GroupFont	字体	GroupGetExternalData	获取外部数据
GroupAlignmentExcel	对齐方式	GroupConnections	连接
GroupNumber	数字	GroupSortFilter	排序和筛选
GroupStyles	样式	GroupDataTools	数据工具
GroupCells	单元格	GroupOutline	分级显示

续表

组　　名	说　　明	组　　名	说　　明
GroupEditingExcel	编辑	GroupProofing	校对
GroupInsertTablesExcel	表格	GroupComments	批注
GroupInsertIllustrations	插图	GroupChangesExcel	更改
GroupInsertChartsExcel	图表	GroupWorkbookViews	工作簿视图
GroupInsertLinks	链接	GroupViewShowHide	显示
GroupInsertText	文本	GroupZoom	显示比例
GroupInsertBarcode	符号	GroupWindow	窗口
GroupThemesExcel	主题	GroupMacros	宏
GroupPageSetup	页面设置	GroupCode	代码
GroupPageLayoutScaleToFit	调整为合适大小	GroupControls	控件
GroupPageLayoutSheetOptions	工作表选项	GroupXml	XML
GroupArrange	排列	GroupModify	修改
GroupFunctionLibrary	函数库	GroupPictureStyles	图片样式
GroupNamedCells	定义的名称	GroupPictureSize	大小
GroupFormulaAuditing	公式审核		

本例案例文件请参考：..\ 第 13 章\ 13-4 创建新组.xlsm

13.2.5　创建对话框启动器：dialogBoxLauncher

【语法】

```
<dialogBoxLauncher>
 <button id="AA" label="BB" screentip="CC" supertip="DD" onAction="EE"
keytip="FF"/>
</dialogBoxLauncher>
```

dialogBoxLauncher 代表对话框启动器，它总是和“<button/>”一起使用，因为对话框启动器只是一个容器，需要在其中放置一个按钮才能发挥作用。

对话框启动器中按钮的 screentip 参数表示屏幕提示；supertip 表示更详细的提示内容；onAction 参数则表示在单击对话框启动器时需要执行的 Sub 过程。

【案例】在“Excel 精灵”选项卡的新组中创建一个对话框启动器，在单击此启动器时可执行名为“工资条设计”的宏，同时需要为对话框启动器指定屏幕提示信息。

【代码】

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
    <tabs>
      <tab id="NewTab" visible="true" label="Excel 精灵" insertAfterMso="TabHome"
keytip="B">
          <group id="Group1" visible="true" label="财务工具">
              <dialogBoxLauncher>
                <button id="dialogOne" screentip="工资条工具" supertip="单击可将工资明细
表转换成工资条" onAction="wage" keytip="G"/>
            </dialogBoxLauncher>
          </group>
        </tab>
      </tabs>
    </ribbon>
</customUI>
```

位于对话框启动器中的命令按钮所关联的 Sub 过程为“工资条设计”，加速键为 G，即用

【Alt+B+G】组合键可以执行此过程。

图 13-11 是对话框启动器外观，图 13-12 是按下【Alt】键后产生的选项卡的加速键提示，图 13-13 是继续按下【B】键后产生的对话框启动器的加速键提示（在 Excel 2016 中不会显示空白的组，因此本例采用 Excel 2010 界面截图，只要在组中放一个按钮就能看到组）。

【效果】

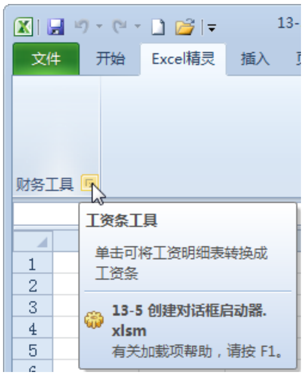


图 13-11　对话框启动器外观

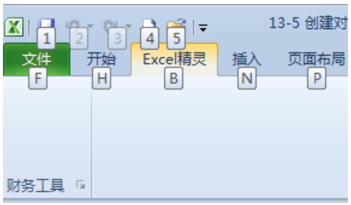


图 13-12　选项卡的加速键

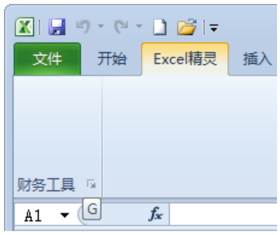


图 13-13　对话框启动器的加速键

【回调】

将 Sub 过程关联到功能区中的按钮和将 Sub 过程关联到传统菜单的方法不同。前者需要使用回调参数，否则模块中的 Sub 过程无法被功能区代码所识别。

功能区的各种控件都有自己独特的回调参数，同时由于功能区中的控件种类繁多，很难记忆这些参数。好在 Custom UI Editor 软件支持自动生成 Sub 过程的外壳，其中包含所有参数代码。

对于本案例，在 Custom UI Editor 软件中编辑完成创建对话框启动器的代码后，单击工具栏的“Generate Callbacks”按钮即可看到以下包含参数的回调过程外壳。

```
'Callback for dialogOne onAction
Sub wage(control As IRibbonControl)
End Sub
```

此过程外壳对应于对话框启动器的命令按钮 button，当单击对话框启动器时会执行此 Sub 过程。

不过此代码并非存放在 Custom UI Editor 软件中，可以在保存并关闭 Custom UI Editor 软件后进入工作簿的 VBE 界面，将以上代码复制到 VBE 界面的模块。

由于 Custom UI Editor 软件只能产生程序外壳，所以需要根据实际需求手动补充代码。

为了简单地展示操作结果，本例采用以下简码：

```
Sub wage(control As IRibbonControl) '放置位置：模块中
    MsgBox "创建工资条.....请补充代码！", vbOKOnly, "友情提示"
End Sub
```

在模块中输入以上代码后，进入工作表界面单击刚创建的对话框启动器，即可得到如图 13-14 所示的效果，表示 Sub 过程与对话框已关联成功。

本例案例文件请参考：..\ 第 13 章\ 13-5 创建对话框启动器.xlsm

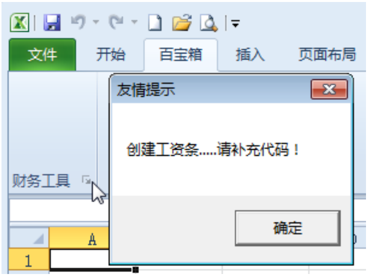


图 13-14　单击对话框启动器

注意：在 Sub 过程的过程名称带参数的情况下不能使用【F8】键逐步运行，所以在编写代码时尽量不要使用参数，调试完成后再补充参数。另外，Custom UI Editor 软件在生成回调参数时不支持汉字，所以在对 onAction 参数赋值时尽量采用字母。

13.2.6　在组中添加命令按钮：button

【语法】

```
<button id="AA" label="BB" visible="CC" enabled="DD" imageMso="EE" size="FF" onAction="GG" screentip="HH" supertip="II" keytip="JJ"/>
```

语法列表中的 enabled 参数用于控制按钮是否处于可用状态，当赋值为“true”或者忽略此参数时，表示可用；当赋值为“false”时，表示不可用；imageMso 参数表示为按钮指定一个内置的图标，如果需要采用外置的图标，则将参数名称替换为“image”，然后赋值为外置图标的名称；size 参数用于控制按钮图标的大小，当赋值为“large”时表示此按钮显示为大图标，当赋值为“normal”或者忽略此参数时将显示为小图标。

命令按钮的 label、visible、enabled、imageMso、size、onAction、keytip、screentip、supertip 都是可选参数。不过在实际工作中，label 和 onAction 两个可选参数都需要赋值，否则无法使用此按钮。

命令按钮放在不同的地方有不同的语法，本语法仅适用于放在组中的命令按钮。

【案例】在“Excel 精灵”选项卡中创建一个名为“创建工资条”的命令按钮，其图标为内置的大图标“ControlLayoutStacked”，加速键为 C；再创建一个名为“个人所得税”的命令按钮，用外置图片作为按钮的图标。

【代码】

（1）先准备一个 ico 格式的图标文件，例如如图 13-15 所示的图标文件。

（2）在 Custom UI Editor 软件中打开需要添加功能区按钮的 xlsm 文件。

（3）选择菜单“Insert”→“Office 2007 Custom UI Part”从而插入一个 customUI.xml 文件。

（4）单击工具栏第 3 个按钮，将图片文件插入到 customUI.xml 文件中，如图 13-16 所示。



图 13-15　图标文件

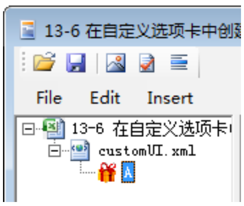


图 13-16　将图标插入到功能区代码文件中

（5）在右边的代码窗口中录入以下代码。

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
 <ribbon startFromScratch="false">
  <tabs>
   <tab id="NewTab" visible="true" label="Excel 精灵" insertAfterMso="TabHome" keytip="B">
     <group id="Group1" visible="true" label="财务工具">
       <button id="button1" label="创建工资条" visible="true" enabled="true"
imageMso="ControlLayoutStacked" size="large"  supertip="功能说明……"  onAction="wage" keytip="C"/>
       <button id="button2" label="个人所得税" visible="true" enabled="true" image="A" size="large"
onAction="tax" keytip="G"/>
     </group>
   </tab>
  </tabs>
 </ribbon>
</customUI>
```

以上代码可以创建一个选项卡、一个组和两个命令按钮。命令按钮的 size 参数赋值为 “large”，所以将显示为大图标，因而一列只能显示一个按钮。如果将其赋值为 normal 或者忽略此参数时将显示为小图标，每列可以显示三个按钮。

由于命令按钮的 label、visible、enabled、imageMso、size、onAction、keytip、screentip 和 supertip 等属性都是可选参数，所以创建第一个按钮的代码也可以简化为如下形式：

```
<button id="button1" label="创建工资条" imageMso="ControlLayoutStacked" size="large" onAction=" wage
"/>
```

但是在实际工作中应尽量将所有可选参数书写完整，以便维护代码。例如以后需要修改某个参数时直接修改值即可，而不用去查找参数名称，同时也避免添加在参数时输错字母造成代码无法执行。

命令按钮不可以直接放在选项卡中，它需要一个比选项卡低一级的容器，此容器可以是对话框启动器、组或者弹出式菜单。

图 13-17 和图 13-18 分别是大图标和小图标样式。

【效果】

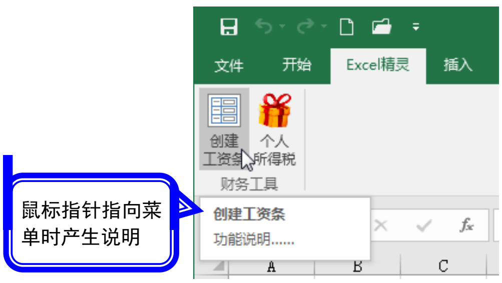


图 13–17　在选项卡中插入大命令按钮

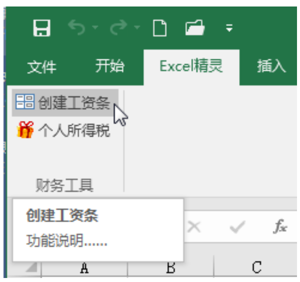


图 13–18　在选项卡中插入小命令按钮

【回调】

单击 Custom UI Editor 软件的 “Generate Callbacks” 按钮将看到以下代码，读者可以将它复制到工作簿的模块中，然后根据需要在过程中添加代码。本章仅用于展示功能区中各种按钮、菜单、复选框和下拉列表等控件的生成方法，省略了 Sub 过程。

```
'Callback for button1 onAction
Sub wage(control As IRibbonControl)
End Sub
'Callback for button2 onAction
Sub tax(control As IRibbonControl)
```

```
End Sub
```

本例案例文件请参考：..\ 第 13 章\ 13-7 在自定义选项卡中创建命令按钮.xlsm

13.2.7　创建切换按钮：toggleButton

【语法】

```
<toggleButton id="AA" label="BB" visible="CC" enabled="DD" imageMso="EE"
size="FF"
onAction="GG" screentip="HH" supertip="II" keytip="JJ"/>
```

切换按钮（toggleButton）只能放在组（group）中，它与同样放在组中的命令按钮的语法完全一样。

【案例】在“Excel 精灵”选项卡中创建一个切换按钮，标题文本为“显示零值”，表示在按下按钮时显示零值，否则隐藏零值。

【代码】

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
 <ribbon startFromScratch="false">
  <tabs>
   <tab id="NewTab" visible="true" label=" Excel 精灵" insertAfterMso="TabHome"
keytip="B">
    <group id="Group1" visible="true" label="视图">
       <toggleButton id="toggleButton1" label="显示零值" visible="true"
enabled="true"  onAction="zero"  imageMso="ChartTypeOtherInsertGallery"
size="large" screentip="零值切换" supertip="按下时显示零值，弹起时不显示零值"
keytip="L"/>
    </group>
   </tab>
  </tabs>
 </ribbon>
</customUI>
```

如果需要添加两个切换按钮，那么两个按钮的 id 绝对不能相同，其他参数允许相同，但是为了使用方便，有必要加以区分，包括对 label 、imageMso、onAction、screentip 和 supertip 等参数赋予不同的值。

切换按钮和命令按钮的区别在于，切换按钮相当于两个命令按钮，单击时执行一个命令，再次单击时执行另一个命令，显示的外观样式也不一样。图 13-19 和图 13-20 分别是按下与弹起时的切换按钮外观。

切换按钮适用于视图切换类需求，可以通过切换按钮展示某类对象的显示状态。

【效果】

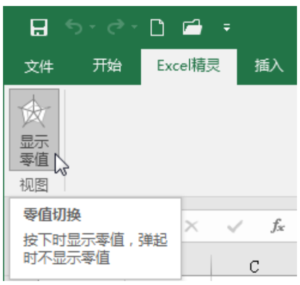


图 13-19　按下状态的切换按钮



图 13-20　弹起状态的切换按钮

【回调】

单击 Custom UI Editor 软件的“Generate Callbacks”按钮将看到以下代码。

```
'Callback for toggleButton1 onAction
Sub zero(control As IRibbonControl, pressed As Boolean)
End Sub
```

参数中的 control 代表切换按钮，pressed 代表按钮的状态，当按下按钮时该值为 True，当按钮弹起时该值为 False。所以可以根据该参数的值来编写对应的 Sub 过程代码，示例如下。

```
Sub zero(control As IRibbonControl, pressed As Boolean) '放置位置：模块中
    '如果切换按钮处于按下状态,那么显示零值，否则不显示零值
      If pressed = True Then    ActiveWindow.DisplayZeros = True Else
ActiveWindow.DisplayZeros = False
End Sub
```

事实上，以上代码也可以简写为如下形式。

```
Sub zero(control As IRibbonControl, pressed As Boolean) '简写形式
    '零值的显示状态由切换按钮的状态决定
    ActiveWindow.DisplayZeros = pressed
End Sub
```

本例案例文件请参考：..\ 第 13 章\ 13-7 创建切换按钮.xlsm

13.2.8 创建弹出式菜单：menu

【语法】

```
<menu id="AA" label="BB" imageMso="CC"   size="DD"   itemSize="EE"
visible="FF" screentip="GG" supertip="HH" keytip="II">
<button id="AA" label="BB" visible="CC" enabled="DD" imageMso="EE"
onAction="FF" screentip="GG" supertip="HH" keytip="II"/>
</menu>
```

弹出式菜单 menu 可通过 size 参数控制图标的显示大小，还可以指定其子菜单的大小，所以既有 size 参数又有 itemSize 参数。后者赋值为“true”表示所有子菜单都显示为大图标，当忽略参数或者赋值为“false”时表示子菜单显示为小图标。

弹出式菜单没有 onAction 参数，如果使用了此参数，那么代码将产生错误。

弹出式菜单的子菜单没有 size 参数，统一通过 itemSize 参数控制大小。

弹出式菜单（menu）和子菜单（button）都可以通过 keytip 参数指定加速键。

弹出式菜单可以有多个子菜单，所以在“<menu>”与“<menu/>”之间可以复制多份“<button/>”，从而产生多个命令按钮，不过要注意 id 不允许重复。

【案例】在“Excel 精灵”选项卡中创建一个弹出式菜单和两个子菜单。

【代码】

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
    <tabs>
      <tab id="NewTab" visible="true" label=" Excel 精灵" insertAfterMso="TabHome"
  keytip="B">
        <group id="Group1" visible="true" label="Excel 精灵">
          <menu id="menu" label="统计工具" imageMso="AutoSum"   size="large"
    itemSize="large">
            <button id="button1" label="多表数据汇总"   imageMso="QueryAppend"   onAction="SheetsQather" />
```

```
            <button id="button2" label="按颜色汇总"   imageMso=" AppointmentColorDialog"
onAction="ColorQather" />
          </menu>
         </group>
        </tab>
       </tabs>
     </ribbon>
   </customUI>
```

要注意按钮“<button/>”必须放在“<menu>”与“</menu>”之间，并且不能使用 size 参数，否则代码会出错。

在本例中对 size 和 itemSize 参数都赋值为“large”，所以弹出式菜单和子菜单都显示为大图标，效果如图 13-21 所示。如果删除这两个参数将得到如图 13-22 所示的效果。

【效果】

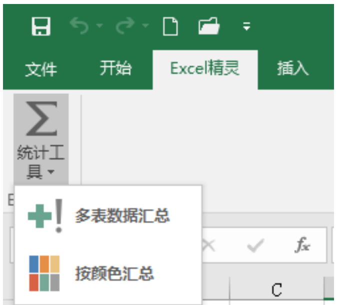


图 13–21　弹出式菜单与两个子菜单

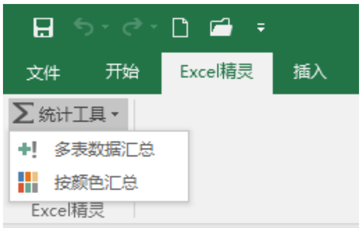


图 13–22　将弹出式菜单和子菜单都显示为小图标

【回调】

单击 Custom UI Editor 软件的“Generate Callbacks”按钮将看到以下代码，读者可以将它复制到模块中，然后根据需求对过程添加代码，本节重点在于创建功能区中各类控件。

```
'Callback for button1 onAction
Sub SheetsQather(control As IRibbonControl)
End Sub
'Callback for button2 onAction
Sub ColorQather(control As IRibbonControl)
End Sub
```

本例案例文件请参考：..\ 第 13 章\ 13-11 创建弹出式菜单.xlsm

13.2.9　创建下拉列表：dropDown

【语法】

```
<dropDown id="AA"  showLabel="BB"  label="CC"  onAction="DD"  enabled="EE">
  <item id="AA" label="BB"  imageMso="CC" />
  <item id="AA" label="BB"  imageMso="CC" />
</dropDown>
```

语法表中的 dropDown 表示创建下拉列表，它的 label 参数表示在下拉列表旁边显示的文字；showLabel 参数用于控制是否显示该文字，当赋值为“false”时可隐藏文字，onAction 参数决定下拉列表关联的宏，它和弹出式菜单完全不同。弹出式菜单是每个子菜单都关联一个宏，而下拉列表控件只有一个宏，关联到 dropDown 自身，其列表中的子元素没有 onAction 参数。

“<item>”语句用于创建列表项目，可对列表项目设置显示的文字标签以及图标。若需要调

用内置图标就改用 imageMso 参数，若调用自定义的图片文件则用 image 参数。

【案例】在“开始”选项卡中创建一个名为“定位”的下拉列表，列表中包含“错误值”、“空单元格”、“公式”、“负数”和“可见单元格”。

【代码】

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
 <ribbon startFromScratch="false">
  <tabs>
     <tab idMso="TabHome">
     <group id="Group1" visible="true" label="Excel 精灵">
      <dropDown id="dropDown1"  showLabel="true"  label="定位"  onAction= "locate"
enabled="true">
       <item id="错误值" label="错误值"  imageMso= "FunctionsDateTimeInsertGallery" />
       <item id="空单元格" label="空单元格"  imageMso=" FunctionsFinancialInsertGallery" />
       <item id="公式" label="公式"  imageMso="FunctionsLogicalInsertGallery" />
       <item id="负数"label="负数"imageMso= "FunctionsLookupReferenceInsertGallery" />
       <item id="可见单元格" label="可见单元格"  imageMso="FunctionsTextInsertGallery" />
      </dropDown>
     </group>
   </tab>
  </tabs>
 </ribbon>
</customUI>
```

代码中“<tab idMso="TabHome">”表示“开始”选项卡，所以它后面的“<group>”命令所创建的组将显示在“开始”选项卡中。同时，由于未指定新组的位置，所以将自动放在“开始”选项卡的最右端，效果如图 13-23 所示。

【效果】

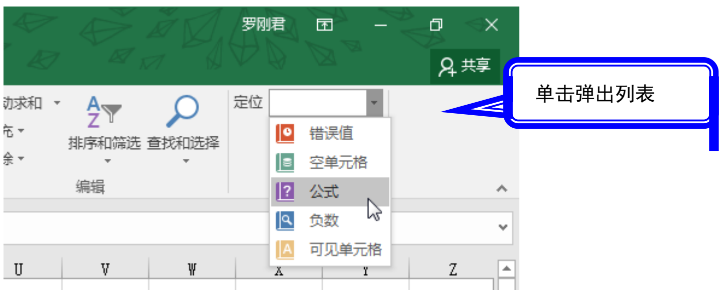


图 13–23 下拉列表

【回调】

单击 Custom UI Editor 软件的“Generate Callbacks”按钮将看到以下代码。

```
'Callback for dropDown1 onAction
Sub locate(control As IRibbonControl, id As String, index As Integer)
End Sub
```

其中参数 control 代表下拉列表控件，id 代表用户从列表中选择项目的 id，例如在选择第二项时返回“空单元格”。参数 index 代表用户选择的下拉列表项目的索引号，从 0 开始，所以当用户选择列表中第一项时该参数返回值为 0。

通过以下 Sub 过程可以更深入地了解各参数含义。

```
Sub locate(control As IRibbonControl, id As String, index As Integer)
 MsgBox "您已选择了列表框控件 " & control.id & " 的第" & index + 1 & "个子元素：  "
```

```
    & id
End Sub
```

本例案例文件请参考：..\ 第 13 章\ 13-9 创建下拉列表.xlsm

13.2.10　创建编辑框：editBox

【语法】

```
<editBox id="AA" label="BB" imageMso="CC" sizeString="DD" maxLength="EE"
visible="FF"
showLabel="GG"    onChange="HH" keytip="II" />
```

其中 label 参数表示显示在编辑框旁边的文字；imageMso 参数表示调用内置图标，改用 image 则可以调用自定义的图片文件作图标，它将显示在编辑框的最左端；sizeString 参数用于控制编辑框的宽度，这个宽度是由赋值的字符串的宽度决定的，而不是由赋值的内容决定的，例如赋值为“999”表示编辑框的宽度等于这三个字符所占的宽度；maxLength 参数用于控制编辑框中录入的字符的数量，赋值为 2 则表示在编辑框中输入字符时不能超过 2 位；showLabel 参数用于控制编辑框旁边的标签的显示状态；onChange 参数表示在编辑框中输入值后按【Enter】键时需要执行的 Sub 过程名称，类似于命令按钮的 onAction 参数。

【案例】创建一个“Excel 精灵”选项卡，使其位于“开始”选项卡之前，并在其中创建一个名为“查找”的编辑框。

【代码】

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
  <tabs>
    <tab id="New tab" label="Excel 精灵" insertBeforeMso="TabHome">
      <group id="NewCustom" label="快速查找">
        <editBox id="FindTxt" label="查找" imageMso="ZoomPrintPreviewExcel"
sizeString="9999999999" maxLength="10" visible="true" showLabel="true"
onChange="Click" keytip="R" />
      </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

代码中 sizeString 参数赋值为 10 个 9，表示编辑框的宽度等于 10 个 9 的宽度；maxLength 参数赋值为 10 表示顶多输入 10 位数，虽然此值是数值，但也需要在前后添加半角的双引号。

编辑框控件包含图标、标签和编辑框三项内容，其外观如图 13-24 所示。

当在编辑框中录入的字符超过 10 位时，将会产生如图 13-25 所示的提示。

【效果】

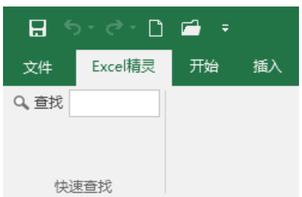


图 13–24　图标、标签与编辑框

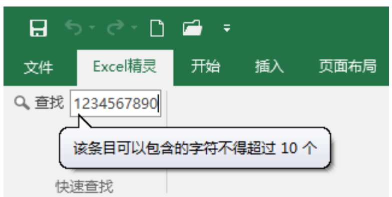


图 13–25　输入超过 10 位字符时产生提示

【回调】

单击 Custom UI Editor 软件的“Generate Callbacks”按钮将看到以下代码。

```
'Callback for FindTxt onChange
Sub Click(control As IRibbonControl, text As String)
End Sub
```

其中参数 control 代表下拉列表控件，text 代表在编辑框中录入的文本。

本例案例文件请参考：..\ 第 13 章\ 13-14 创建编辑框.xlsm

13.2.11 锁定或隐藏内置功能

【语法】

功能区中所有内置的命令都可以锁定，即禁止使用，锁定内置命令的语法如下：

```
<commands>
 <command idMso="AA" enabled = "BB" />
</commands>
```

其中“<commands>”与“<ribbon>”是同级别的对象，所以不能放在“<ribbon>”与“<ribbon/>”之间。

“<command />”代表调用内置命令，通过 idMso 参数指定内置命令的 id 即可。当 enabled 参数赋值为“false”时，表示禁用此内置命令。

【案例】禁用内置的“保存”、“合并居中”、“复制”和“剪切”功能。

【代码】

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
<commands>
 <command idMso="FileSave" enabled = "false" />
 <command idMso="MergeCenterMenu" enabled = "false"   />
 <command idMso="Copy" enabled = "false" />
 <command idMso="Cut" enabled = "false" />
</commands>
</customUI>
```

禁用内置命令直接将“<commands>”与“</commands>”放置在“<customUI>”与“</customUI>”壳中，并且将 enabled 赋值为“false”即可，效果如图 13-26 和图 13-27 所示。

内置命令可以禁用，但是不能隐藏，因为 command 对象是没有 visible 参数的。

不过内置的选项卡和组支持 visible 参数，可以隐藏任意内置组或选项卡。

【效果】

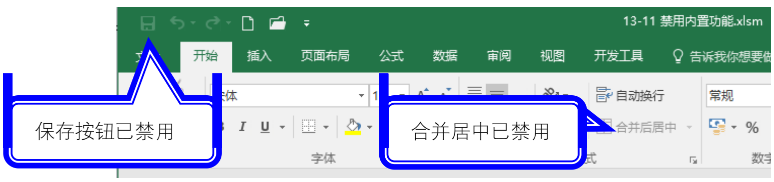


图 13–26 禁用“保存”功能

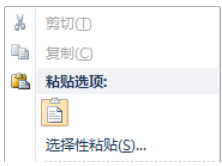


图 13–27 禁用“复制”与“剪切”功能

本例案例文件请参考：..\ 第 13 章\ 13-15 禁用内置功能.xlsm

13.3　使用回调函数强化功能区

回调函数专用于功能区控件，对自定义的功能区控件使用回调函数能强化控件的功能，实现动态获取数据，并根据获取的数据调整控件的状态。

熟悉回调函数后，用户在开发功能区控件领域将呈现另一片天地。

13.3.1　为什么需要使用回调函数

使用回调函数后，自定义功能区控件的代码可以获取单元格中的值或者 Sub 过程中变量的值，从而使自定义的功能区控件更灵活，能满足更多的需求。

例如“13-7 创建切换按钮.xlsm”文件中的 VBA 代码可以读取切换按钮的显示状态，然后根据其状态调整工作表中零值状态。但是这只是单向的控制，不够完美，若切换按钮也能根据当前工作表零值的显示状态调整自己的状态，则更人性化。

简而言之，使用回调函数就是为了方便在 VBA 中控制功能区中某个控件的某个属性。

13.3.2　回调函数详解

回调函数绝大多数是以 get 开头，例如 getLabel、getText 和 getImage 等，表示控件对象的某个参数在后期通过 VBA 的 Sub 过程赋值，而非在前期编写功能区代码时赋值。

当然 onChange 和 onAction 也是回调函数。

所有回调函数的名称及其功能如表 13-3 所示。

表 13-3　回调函数

函　数	功　能	函　数	功　能
getDescription	获取控件描述	getSelectedItemID	获取选中的子项目的ID
getEnabled	确定按钮是否可用	getSelectedItemIndex	获取选中的子项目的序号
getImage	获取图像	getText	获取文本
getKeytip	获取控件的加速键	getItemHeight	获取子项高度
getLabel	获取控件的标签	getItemWidth	获取子项宽度
getScreentip	获取控件的提示	getTitle	获取标题
getSupertip	获取控件的详细提示	getContent	获取XML代码内容
getShowImage	判断控件是否显示图标	GetEnabledMso	获取内置控件的可用性
getShowLabel	判断控件是否显示标签	GetImageMso	获取内置控件的图像
getSize	获取控件的大小	GetLabelMso	获取内置控件的标签
getVisible	获取控件的可见性	GetPressdMso	判断内置控件是否按下
getItemCount	获取子项的数量	GetScreenTipMso	获取内置控件的提示
getItemID	获取子项的ID	GetSuperTipMso	获取内置控件的详细提示
getItemImage	获取子项的图像	GetVisibleMso	获取内置控件的可见性
getItemLabel	获取子项的标签	onChange	指定文本改变时执行的宏名称
getItemScreentip	获取子项的提示	onAction	指定单击按钮时时执行的宏名称
getItemSupertip	获取子项的详细提示		

其中应用最频繁的是以下几个回调函数：getEnabled、getImage、getItemImage 、getLabel、getVisible 和 onAction。

1．getEnabled

getEnabled 函数表示从 VBA 代码中取值，并根据该值决定控件是否可用。getEnabled 函数其实就是动态的 enabled 参数。换言之，一个控件是否处于禁用状态，可以通过回调函数

getEnabled 来决定，它的返回值决定控件是否可用，而非在编写功能区代码时直接赋值为“true”或者“false”。

例如，功能区中的自定义命令按钮的功能是合并单元格的值，那么在使用 getEnabled 后可以在 VBA 代码中判断活动表是否为图表以及 Selection 是图片还是区域，如果活动表是图表或者 Selection 是图片，那么会传递一个逻辑值“false”给 getEnabled 函数，该函数再将“false”传递给命令按钮 button，那么命令按钮将自动显示为禁用状态，从而避免执行命令出错。

以上思路可以用于所有支持回调函数的控件中。

再如某自定义的命令按钮用于汇总上月的生产数据，领导要求每月 4 号之前汇总完毕。在此前提下，可以通过 getEnabled 函数获取当前日期，假设日期是 1 号、2 号、3 号，则此按钮处于可用状态，在这三天可以单击按钮汇总上月的数据；如果日期大于 3 号则按钮自动呈现禁用状态。此思路可以让按钮具有智能，根据需求自动调节。

getEnabled 函数的用法如下。

（1）在编写功能区控件的代码时，用 getEnabled 替换原来的 enabled 参数，然后对参数指定一个 Sub 过程名称，而不是使用逻辑值“true”或“false”。

（2）在 Custom UI Editor 软件中单击工具栏的“Generate Callbacks”按钮即可看到 getEnabled 的回调过程外壳。假设对 getEnabled 函数赋值为“ABC”，那么将产生以下代码。

```
Sub ABC(control As IRibbonControl, ByRef returnedVal)
End Sub
```

其中参数 control 代表当前控件，参数 returnedVal 代表传递给 getEnabled 函数的值，所以在 Sub 过程中根据需求对变量 returnedVal 赋值即可，该值会在显示控件时由 getEnabled 函数传递给控件。

（3）将回调过程的程序外壳复制到 VBE 界面的模块中，然后在该过程中对参数 returnedVal 赋值，该值将在显示控件时自动传递给控件，从而改变控件的显示状态。

所有具有 enabled 参数的控件都支持 getEnabled 函数。

2. getImage

getImage 函数表示从 VBA 代码中取值，并根据该值决定控件的图标名称。getImage 函数其实相当于动态的 image 参数。

和获取 getEnabled 的回调参数的方法一样，当在开发功能区控件的代码中使用了 getImage 函数时，在 Custom UI Editor 软件中单击工具栏的“Generate Callbacks”按钮即可看到 getImage 函数的回调过程外壳。假设对 getImage 函数赋值为“ABC”，那么 getImage 函数对应的回调过程外壳如下：

```
Sub ABC(control As IRibbonControl, ByRef returnedVal)
End Sub
```

参数 control 代表当前控件，参数 returnedVal 代表传递给 getImage 函数的值，所以在 Sub 过程中根据需求对变量 returnedVal 赋值即可，该值会在显示控件时由 getImage 函数传递给控件。

所有具有 image 参数的控件都支持 getImage 函数。

3. getItemImage

getItemImage 函数表示从 VBA 代码中取值，并根据该值决定控件的图标名称。getItemImage 函数其实就是动态的 itemImage 参数。

假设对 getItemImage 函数赋值为“ABC”，那么 getItemImage 函数对应的回调过程外壳如下。

```
Sub ABC(control As IRibbonControl, index As Integer, ByRef returnedVal)
End Sub
```

参数 control 代表当前控件，参数 index 代表控件子元素的索引号，参数 returnedVal 代表传递给 getItemImage 函数的值。

getItemImage 函数仅用于 comboBox、dropDown 和 gallery3 种带有下拉列表的控件。

4．getLabel

getLabel 函数表示从 VBA 代码中取值，并根据该值决定控件显示在功能区中的字符。getLabel 函数其实就是动态的 label 参数。

假设对 getLabel 函数赋值为“ABC”，那么 getLabel 函数对应的回调过程外壳如下。

```
Sub ABC(control As IRibbonControl, ByRef returnedVal)
End Sub
```

参数 control 代表当前控件，参数 returnedVal 代表传递给 getLabel 函数的值。

所有具有 label 参数的控件都支持 getLabel 函数。

5．getVisible

getVisible 函数表示从 VBA 代码中取值，并根据该值决定控件是否可见。getVisible 函数其实就是动态的 visible 参数。

假设对 getVisible 函数赋值为“ABC”，那么 getVisible 函数对应的回调过程外壳如下。

```
Sub ABC(control As IRibbonControl, ByRef returnedVal)
End Sub
```

参数 control 代表当前控件，参数 returnedVal 代表传递给 getVisible 函数的值。

所有具有 visible 参数的控件都支持 getVisible 函数。

13.3.3　创建 1 到 3 号才能使用的按钮

【案例要求】在“开始”选项卡创建一个名为“汇总上月资料”的按钮，在每月的 1 到 3 号打开工作簿时，该按钮呈可用状态，而在其他时间打开工作簿时，该按钮呈禁用状态。

【知识要点】getEnabled。

【操作步骤】

（1）新建一个空白工作簿，然后保存为 xlsm 格式的文件。

（2）打开 Custom UI Editor 软件，并从软件中打开刚才所保存的工作簿。

（3）选择菜单“Insert”→“Office 2007 Custom UI Part”，从而插入一个 customUI.xml 文件，然后在代码窗口中录入以下代码，用于在“开始”选项卡中创建新组及命令按钮。

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
 <ribbon startFromScratch="false">
  <tabs>
  <tab idMso="TabHome">
     <group id="Group1" visible="true" label="汇总">
        <button id="button1" label="汇总上月资料" visible="true" getEnabled="ABC"
imageMso="ControlLayoutStacked" size="large" onAction="Qather" />
     </group>
    </tab>
  </tabs>
```

```
    </ribbon>
  </customUI>
```

（4）单击“Generate Callbacks”按钮，产生 getEnabled 与 onAction 两个回调函数所对应的过程外壳，将其复制，然后保存并关闭 Custom UI Editor 软件。

（5）使用 Excel 打开之前创建的工作簿文件，使用【Alt+F11】组合键打开 VBE 窗口。

（6）选择菜单“插入”→“模块”，然后将刚才复制的两段代码粘贴到模块中。

（7）对名为 ABC 的过程添加赋值语句，最终代码如下（过程 Qather 的代码省略，本例主要展示回调函数 getEnabled 的应用思路）。

```
Sub ABC(control As IRibbonControl, ByRef returnedVal)   '放置位置：模块中
  If Day(Date) >= 1 And Day(Date) <= 3 Then             '如果打开文件的日期大于等于 1 而且小于等于 3
      returnedVal = True                                '将变量赋值为 true
  Else                                                  '否则
      returnedVal = False                               '将变量赋值为 false
  End If
End Sub
```

（8）保存工作簿并重启，如果当前日期不属于 1 号、2 号或者 3 号，那么按钮将呈禁用状态，如图 13-28 所示。

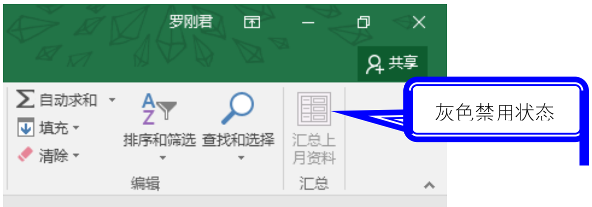


图 13–28　禁用状态的命令按钮

【知识补充】

（1）在过程“ABC”中，首先利用代码“Day(Date)”计算打开工作簿的日期，如果处于 1 到 3 号，那么对 returnedVal 变量赋值为“True”，否则赋值为“False”，此值将传递给 getEnabled 函数，getEnabled 函数再根据此值决定命令按钮是否可用。

（2）在定制功能区的代码中对 enabled 参数赋值时采用“"false"”或者“"true"”，它是一个所有字母都小写的文本字符串，但是在回调过程对 getEnabled 函数赋值时需要采用逻辑值“False”或者“True”，不能添加双引号，也不区分大小写。

本例案例文件请参考：..\ 第 13 章\13-16 1 到 3 号能使用的命令按钮.xlsm

13.3.4　创建按下与弹起时自动切换图标的按钮

【案例要求】在“视图”选项卡中创建一个“显示零值”的命令按钮，当按钮被按下时显示零值，按钮的图标显示为勾；当按钮弹起时隐藏零值，按钮的图标显示为叉。

【知识要点】getImage 和 onLoad。

【操作步骤】

（1）新建一个空白工作簿，然后保存为 xlsm 格式的文件。

（2）打开 Custom UI Editor 软件，并从软件中打开刚才所保存的工作簿。

（3）选择菜单“Insert”→“Office 2007 Custom UI Part”从而插入一个 customUI.xml 文件，

然后在代码窗口中录入以下代码，用于在“视图”选项卡中创建新组及切换按钮。

```
<customUI xmlns=http://schemas.microsoft.com/office/2006/01/customui
onLoad="Intialize">
  <ribbon startFromScratch="false">
    <tabs>
      <tab idMso="TabView">
        <group id="Group1"   label="零值控制"   insertAfterMso="GroupViewShowHide">
            <toggleButton id="toggleButton1" label="显示零值" visible="true"
enabled="true"   onAction="zero"   getImage="getImage" size="large"
screentip="零值切换" supertip="按下时显示零值，弹起时不显示零值" keytip="L"/>
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

要注意第一句代码的末尾加了“onLoad="Intialize"”，表示调用此代码时先执行名为“Intialize”的 Sub 过程。

另外“<tab>”语句采用了“idMso="TabView"”，表示当前对象是“视图”选项卡。

在对切换按钮 toggleButton 指定图标时采用了“getImage="getImage"”，表示按钮的图标通过名为“getImage”的 Sub 过程来指定。

（4）单击“Generate Callbacks”按钮产生 Intialize、zero 与 getImage 三个回调过程所对应的过程外壳，将它们复制，然后保存并关闭 Custom UI Editor 软件。

（5）用 Excel 打开之前创建的工作簿文件，使用【Alt+F11】组合键打开 VBE 窗口。

（6）选择菜单“插入”→“模块”，然后将刚才所复制的三段代码粘贴到模块中。

（7）对三个过程添加代码，使其能正常工作，最终代码如下。

```
Dim bl As Boolean        '声明一个公共变量，此变量代码切换按钮的状态
Dim rib As IRibbonUI     '声明一个公共变量，IRibbonUI 代表一个 Ribbon 类的对象实例
Sub Intialize(ribbon As IRibbonUI)
'将 IRibbonUI 类赋予变量 rib，从而载入缓存供以后调用
'通常是通过 Invalidate 或者 InvalidateControl（controlID）方法调用，从而更新全部或者某个控件的值
  Set rib = ribbon
End Sub
```

以上过程将在启动工作簿时执行，作用是将功能区对象 Ribbon 赋予变量 rib，即载入缓存中，供其他代码随时调用。

```
'单击切换按钮时需要调用的过程，参数 pressed 代表按钮的状态
Sub zero(control As IRibbonControl, pressed As Boolean)
    '零值的显示状态由切换按钮的状态决定
    ActiveWindow.DisplayZeros = pressed
    '将按钮的状态赋值给变量 bl,然后在“getImage”过程中根据 bl 的值决定按钮使用何种图标
    bl = pressed
    '强制更新功能区，否则不会执行过程“getImage”，不执行过程就不能载入图标
  rib.Invalidate
End Sub
```

此过程对应于切换按钮的 onAction 参数，即单击切换按钮时可执行此过程。过程中的 pressed 参数代表按钮的状态。在单击按钮时 VBA 会将按钮的状态传递给变量 pressed，而过程中的代码又将此变量的值传递给公共变量 bl，从而使后面的过程“getImage”可以接收此变量的值，然后根据变量的值决定为切换按钮指定何种图标。公共变量 bl 是过程“zero”与“getImage”

的中转站，如果没有这个公共变量，过程“getImage”就无法识别切换按钮的当前状态，从而也无法正确地对切换按钮的图标属性赋值。

```
'为切换按钮指定图标，其中参数 returnedVal 的值将传递给 getImage 函数
'getImage 函数再将值传递给切换按钮，传递完成后按钮会更新屏幕上的显示图标
Sub getImage(control As IRibbonControl, ByRef returnedVal)
    If bl = False Then                      '如果公共变量 bl 的值等于 false
        returnedVal = "DeclineInvitation"   '那么对参数 returnedVal 赋值为 DeclineInvitation
    Else                                    '否则
        returnedVal = "AcceptInvitation"    '对参数 returnedVal 赋值为 AcceptInvitation
    End If
End Sub
```

此过程表示接收到公共变量 bl 的值后，根据变量的值决定参数 returnedVal 的值。而 returnedVal 参数的值会传递给切换按钮，从而决定切换按钮的图标。

（8）保存并重启工作簿，然后进入“视图”选项卡，单击“显示零值”按钮，按钮将呈现按下状态，同时按钮的图标显示为勾，如图 13-29 所示，再次单击按钮时，工作表中的所有零值都会被隐藏起来，同时按钮的图标更新为叉，如图 13-30 所示。

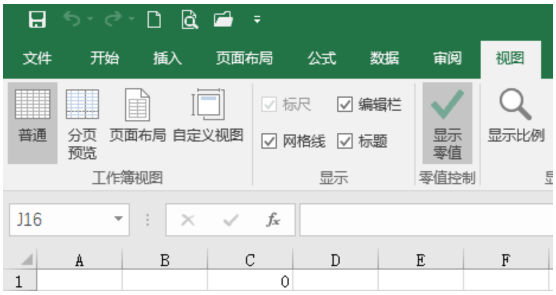


图 13–29　按钮按下时显示的图标

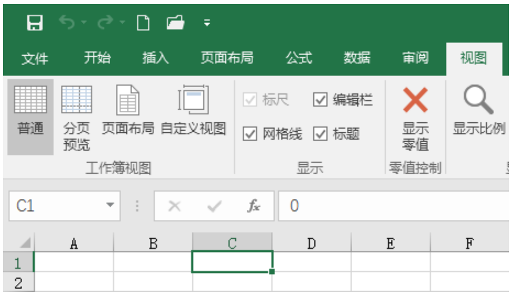


图 13–30　按钮弹起时显示的图标

【知识补充】

（1）在单击切换按钮时，切换按钮的状态会传递给“zero”过程的 pressed 参数，但是不会传递给“getImage”过程，所以需要使用一个公共变量 bl 来做中转站，在过程“zero”中赋值，再在过程“getImage”中接收值。

（2）过程“Intialize”的作用是将功能区 IRibbonUI 赋予变量 rib，从而载入缓存供“rib.Invalidate”语句调用，没有过程“Intialize”就不能更新功能区，过程“getImage”中的 returnedVal 的值就不能传递给切换按钮。

（3）本例重点在于根据按钮的状态修改按钮的图标，VBA 中的 Sub 过程并不完善。例如当打开工作簿时，不管工作表中是否显示零值，按钮都默认为弹起状态，图标为叉。也就是说，切换按钮的图标与状态并没有与零值的状态同步，只有单击按钮后才同步。这将作为思考题放在本章末尾，请读者自行完善代码。当然，在本书的随书案例文件中会有答案。

本例案例文件请参考：..\ 第 13 章\ 13-13 创建按下与弹起时自动切换图标的按钮.xlsm

13.3.5　在功能区中快速查找

【案例要求】在“开始”选项卡中创建一个下拉列表和一个编辑框，下拉列表决定查找方式，编辑框决定查找内容，两者结合可实现快速定位目标单元格。

【知识要点】onChange。

【操作步骤】

（1）新建一个空白工作簿，然后保存为 xlsm 格式的文件。

（2）打开 Custom UI Editor 软件，并从软件中打开刚才所保存的工作簿。

（3）选择菜单“Insert”→“Office 2007 Custom UI Part”从而插入一个 customUI.xml 文件，然后在代码窗口中录入以下代码，用于在“开始”选项卡创建新组及下拉列表、编辑框。

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
  <tabs>
    <tab idMso="TabHome">
     <group id="group1" label="快速查找" insertBeforeMso="GroupFont">
        <dropDown id="Style" showLabel="true" label="匹配方式" onAction
="dropDownChange" >
          <item id="xlWhole" label="xlWhole" imageMso="WatchWindow" />
          <item id="xlPart" label="xlPart" imageMso="ZoomPrintPreviewExcel" />
        </dropDown>
        <editBox id="FindTxt" label="查找内容" imageMso="ZoomPrintPreviewExcel"
sizeString="999999999999" maxLength="30" visible="true" showLabel="true"
onChange="editBoxChange" keytip="R" />
     </group>
     </tab>
    </tabs>
  </ribbon>
</customUI>
```

（4）单击“Generate Callbacks”按钮产生 dropDownChange 与 editBoxChange 两个回调过程所对应的过程外壳，将它们复制，然后保存并关闭 Custom UI Editor 软件。

（5）用 Excel 打开前面创建的工作簿文件，使用【Alt+F11】组合键打开 VBE 窗口。

（6）选择菜单“插入”→“模块”，然后将刚才复制的两段代码粘贴到模块中。

（7）对两个过程添加代码，使其能正常工作，最终代码如下。

```
Public str As String                                  '声明一个公共变量（放置位置：模块中）
'单击下拉列表控件时执行的过程，有三个参数，第一参数代表控件本身，第二参数代表选择的子项的 id
'第三参数代表选择的子项的序号
Sub dropDownChange(control As IRibbonControl, id As String, index As Integer)
    str = id                                          '将选择的子项的 id 传递给变量 Str
End Sub
'单击文字框时执行的过程。有两个参数，第二参数代表文字框中显示的文本
Sub editBoxChange(control As IRibbonControl, text As String)
    If Len(text) = 0 Then Exit Sub                    '如果文字框空白则中断程序
    If str = "" Then MsgBox "请设置 lookat 参数": Exit Sub '如果公共变量空白则产生提示并且结束过程
    Dim FirstCell As Range, Rng As Range              '声明变量
    With Cells
        '执行查找，匹配方式由变量 Str 决定
        Set FirstCell = .Find(text, LookIn:=xlValues, lookat:=IIf(str = "xlWhole", xlWhole, xlPart))
        If Not FirstCell Is Nothing Then              '如果已找到
            firstAddress = FirstCell.Address          '记录第一个目标的位置
            Do                                        '循环查找其他目录
                If Rng Is Nothing Then Set Rng = FirstCell Else Set Rng =Union(Rng, FirstCell)
                Set FirstCell = .FindNext(FirstCell) '查找下一个
            Loop While FirstCell.Address <> firstAddress
```

```
            End If
        End With
        If Not Rng Is Nothing Then                        '如果找到
            Rng.Select                                    '选择所有对象
        Else                                              '否则提示未找到
            MsgBox "Sorry,未找到 “" & text & "” "
        End If
End Sub
```

在以上代码中，首先声明了一个公共变量 str，它作为两个 Sub 过程的中转站传递下拉列表控件的值给编辑框，编辑框接收到值后根据该值决定搜索方式。至于搜索的内容则由编辑框的 text 参数决定。

（9）保存并重启工作簿，在“开始”选项卡中将看到如图 13-31 所示的控件外观。

（10）在编辑框中录入字符“T”并按【Enter】键，此时 Excel 将弹出如图 13-32 所示的提示框，表示未设置匹配方式。

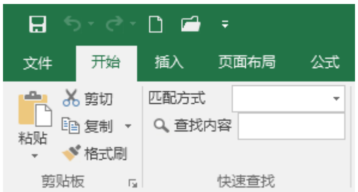


图 13–31　控件外观

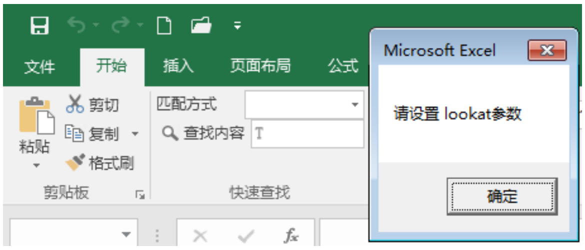


图 13–32　查找前未设置匹配方式

（11）将匹配方式设置为 xlPart，然后在编辑框中录入字符“赵”并按【Enter】键，程序会瞬间选中所有包括“赵”的单元格，如图 13-33 所示。

（12）将匹部方式设置为 xlWhole，并在编辑框中录入字符“赵文强”，然后按下【Enter】键，程序会弹出如图 13-34 所示的提示框，表示按精确匹配方式查找时未找到目标。

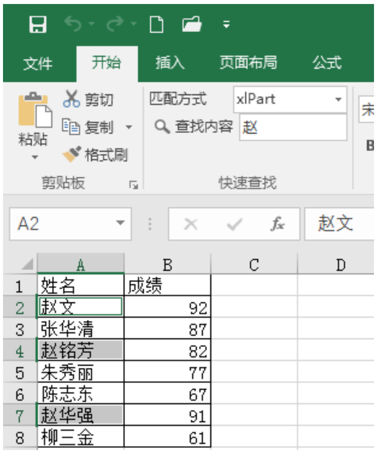


图 13–33　按糊模匹配方式查找赵

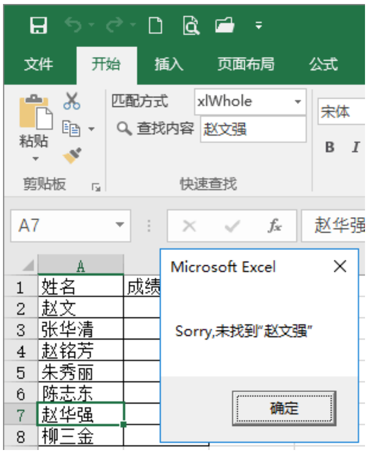


图 13–34　按精确匹配方式查找赵文强

【知识补充】

（1）在本例中执行查找任务时，需要使用下拉列表的值和编辑框的值，由于编辑框无法获得下拉列表的值，所以先声明一个公共变量，在下拉列表的“dropDownChange”过程中将值传递

给变量，然后在编辑框中读取该变量的值。

（2）下拉列表的两个项目使用了 xlWhole 和 xlPart，也可以修改为汉字，例如“精确匹配”和“模糊匹配”，修改功能区代码后 Sub 过程也需要同步修改。

本例案例文件请参考：..\ 第 13 章\ 13-14 通过编辑框执行精确查找.xlsm

13.3.6 在组的标签处显示日期及问候语

【案例要求】修改前一案例的功能区代码，使“快速查找”组的文字（label 属性）显示为“您好”及当前日期。

【知识要点】getLabel。

【操作步骤】

（1）新建一个空白工作簿，然后保存为 xlsm 格式的文件；

（2）打开 Custom UI Editor 软件，并从软件中打开刚才保存的工作簿；

（3）选择菜单“Insert”→“Office 2007 Custom UI Part”从而插入一个 customUI.xml 文件，然后在代码窗口中录入以下代码，用于在“开始”选项卡中创建新组及命令按钮。

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
  <tabs>
    <tab idMso="TabHome">
     <group id="group1" getLabel="getLabel" insertBeforeMso="GroupFont">
        <dropDown id="Style" showLabel="true" label="匹配方式"
onAction="dropDownChange" >
         <item id="xlWhole" label="xlWhole" imageMso="WatchWindow" />
         <item id="xlPart" label="xlPart" imageMso="ZoomPrintPreviewExcel" />
       </dropDown>
       <editBox id="FindTxt" label="查找内容" imageMso="ZoomPrintPreviewExcel"
sizeString="999999999999" maxLength="30" visible="true" showLabel="true"
onChange="editBoxChange" keytip="R" />
     </group>
     </tab>
    </tabs>
  </ribbon>
</customUI>
```

代码中的组 group 使用了回调函数 getLabel，表示通过 VBA 代码后期指定组的文字。

（4）单击“Generate Callbacks”按钮产生 getLabel、dropDownChange 和 editBoxChange 三个回调过程所对应的过程外壳，复制 getLabel 的回调过程，然后保存并关闭 Custom UI Editor 软件。

（5）用 Excel 打开前面创建的工作簿文件，使用【Alt+F11】组合键打开 VBE 窗口。

（6）选择菜单“插入”→“模块”，然后将刚才所复制的代码粘贴到模块中。

（7）对过程添加代码，使其能正常工作，最终代码如下。

```
'为组添加文字，包括“您好”及当前日期和星期
Sub getLabel(control As IRibbonControl, ByRef returnedVal) '放置位置：模块中
 returnedVal = "您好  " & Format(Date, "yyyy-mm-dd AAAA")
End Sub
```

（8）保存并重启工作簿，在“开始”选项卡中的新组将显示包含日期、星期的问候语，如图 13-35 所示。

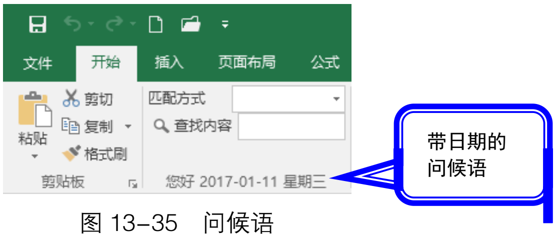


图 13–35 问候语

【知识补充】

（1）本例是回调函数 getLabel 的又一应用，借助回调函数使功能区中自定义的控件更人性化，满足更多需求。

（2）本例只要求显示日期，所以过程较简单，不需要更新内容。如果要求显示时、分、秒，且实时更新，那么除修改 Format 函数的第二参数外，还须要使用 Application.OnTime 方法创建计划任务。

本例案例文件请参考：..\ 第 13 章\ 13-15 在组的标签处显示日期及问候语.xlsm

13.3.7 调用大图片创建下拉菜单

图片库是功能区独有的新功能，它可以实现实时预览，能极大提升工作效率，专用名词称之为 gallery。

gallery 也是功能区的基本控件之一，不过由于它的应用比较复杂，必须配合回调函数才能使用，所以放在本章末尾，

gallery 的语法如下。

```
<gallery id="AA" label="BB" size="CC" showLabel="DD" image="EE"
columns="FF" rows="GG" itemHeight="HH" itemWidth="II" supertip="JJ" getItemCount="KK" getItemID="LL"
getItemImage="MM" getItemSupertip="NN" getItemLabel="OO" onAction="PP"/>
```

库的参数有近 20 个，相对于其他控件复杂得多，不过其中大多是可选参数，在实际使用时不需要对每个参数都赋值。

参数中的 showLabel 用于控制是否显示标签，有 true 和 false 两个选项；columns 和 rows 分别代表库的显示方式，表示由几行、几列组成；itemHeight 和 itemWidth 参数分别代表子项目的显示高度和宽度；getItemCount、getItemID、getItemImage、getItemSupertip 和 getItemLabel 分别用于指定子项目的数量、id、图标、提示和标签。

接下来通过案例展示 gallery 的制作过程。

【案例要求】在“开始”选项卡中添加一个库 gallery，库的子菜单包含“合并居中”和“取消合并”，分别采用两张大照片作图标，从而实现能够根据菜单图标预览菜单功能。

【知识要点】gallery、getItemCount、getItemID、getItemImage、getItemSupertip 和 getItemLabel。

【操作步骤】

（1）新建一个空白工作簿，然后保存为 xlsm 格式的文件。

（2）打开 Custom UI Editor 软件，并从软件中打开刚才保存的工作簿。

（3）单击菜单“Insert”→“Office 2007 Custom UI Part”从而插入一个 customUI.xml 文件，然后在代码窗口中录入以下代码，用于在“开始”选项卡创建新组及库。

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
 <ribbon>
```

```
    <tabs>
      <tab idMso="TabHome">
        <group id="group1" label="合并单元格" insertAfterMso="GroupClipboard">
          <gallery id="gallery1" label="合并单元格" size="large" showLabel="true"
  imageMso="TableStyleClear"  columns="1" rows="2" itemHeight="157" itemWidth= "142" supertip="合并单元格且保留所有数据" getItemCount="getItemCount" getItemID="getItemID" getItemImage="getItemImage" getItemSupertip=" getItemSupertip" getItemLabel="getItemLabel" onAction= "Action"/>
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

以上代码可以在“开始”选项卡中创建一个新组，组中包含一个图片库 gallery，库的行数为 2，列数为 1，每行的高度为 157，宽度为 142。

其中最重要的是使用了 getItemCount、getItemID、getItemImage、getItemSupertip 和 getItemLabel 等 5 个回调函数，这些回调函数可以在后期指定图片的路径、id、数量提示和显示字符等属性。

（4）单击“Generate Callbacks”按钮产生 getItemCount、getItemImage、getItemLabel 和 Action 四个回调过程所对应的过程外壳。事实上应该有 6 个才对，还需要包括 getItemSupertip 和 getItemID，由于未知原因并未一并罗列出来，笔者在此将其补充完整。

```
'Callback for gallery1 getItemCount
Sub getItemCount(control As IRibbonControl, ByRef returnedVal)
End Sub
'Callback for gallery1 getItemImage
Sub getItemImage(control As IRibbonControl, index As Integer, ByRef
returnedVal)
End Sub
'Callback for gallery1 getItemLabel
Sub getItemLabel(control As IRibbonControl, index As Integer, ByRef
returnedVal)
End Sub
'Callback for gallery1 onAction
Sub Action(control As IRibbonControl, id As String, index As Integer)
End Sub
'Callback for gallery1 getItemID
Sub getItemID(control As IRibbonControl, index As Integer, ByRef id)
End Sub
'Callback for gallery1 getItemSupertip
Sub getItemSupertip(control As IRibbonControl, index As Integer, ByRef
supertip)
End Sub
```

保存并关闭 Custom UI Editor 软件。

（5）准备两个图片文件，分别用于“合并居中”与“取消合并”两个功能按钮的图标，从而使用户可以在执行代码之前预览效果，如图 13-36 所示。

两个图片尽量调整为宽度 142、高度 157，与代码中指定的高度与宽度一致，然后将图片存放在与当前工作簿相同的路径中。

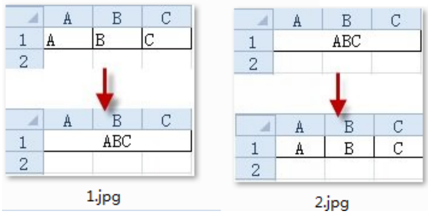


图 13–36　将要显示在库中的图片

（6）用 Excel 打开前面创建的工作簿文件，使用【Alt+F11】组合键打开 VBE 窗口。

（7）选择菜单“插入”→“模块”，然后将上述 6 段代码复制到模块中。

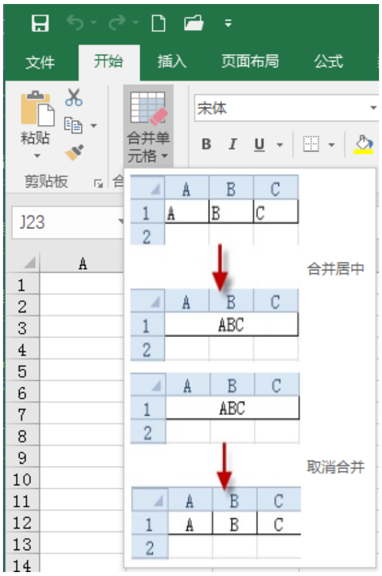
（8）对 6 个过程添加代码，使其能正常工作，最终代码如下。

```
'加载同路径下的 jpg 图片　　（放置位置：模块中）
'图片名称必须是 1.jpg 和 2.jpg,与当前文件放在同一个路径下
Sub getItemImage(control As IRibbonControl, Index As Integer, ByRef
returnedVal)
    Set returnedVal = LoadPicture(ThisWorkbook.Path & "\" & Index + 1 & ".jpg")
End Sub
'将库的子项目数量设置为 2
Sub getItemCount(control As IRibbonControl, ByRef returnedVal)
    returnedVal = 2
End Sub
'指定库的每一个子项的 ID
Sub getItemID(control As IRibbonControl, Index As Integer, ByRef Id)
    Id = Index + 1
End Sub
'指定每个子项目的提示信息，由于项目有多个，需要使用数组
Sub getItemSupertip(control As IRibbonControl, Index As Integer, ByRef
supertip)
    supertip = Array("将数组合并居中，但是保留所有合并前的数据", "取消合并居中，还原
合并前的状态，不丢失数据")(Index)
End Sub
'指定图片右边显示的文本标签
Sub getItemLabel(control As IRibbonControl, Index As Integer, ByRef
returnedVal)
    returnedVal = Array("合并居中", "取消合并")(Index)
End Sub
'单击库的子项时执行的宏过程
Sub Action(control As IRibbonControl, Id As String, Index As Integer)
Call 合并居中(Index)
End Sub
```

事实上过程“Action”调用了另一个名为“合并居中”的过程，从而完成“合并居中”与“取消合并”两个功能。本例重点展示库的应用，而且由于过程“合并居中”的代码较长，读者可以从随书案例文件中获取该过程源代码。

（9）保存并重启工作簿，在“开始”选项卡中将看到创建的新组，组中包含一个名为“合并单元格”的库 gallery。单击“合并单元格”将弹出库的两个子项目，分别为“合并居中”和“取消合并”。从功能上讲，库 gallery 与弹出式菜单 menu 极为相似，不过库 gallery 更人性化，可以预览效果，如图 13-37 所示。

当鼠标指针指向库的子元素时还提供屏幕提示，如图 13-38 所示。



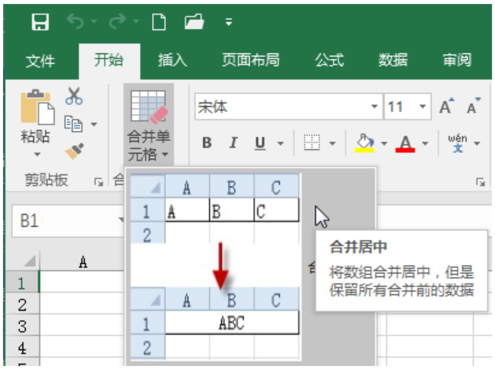
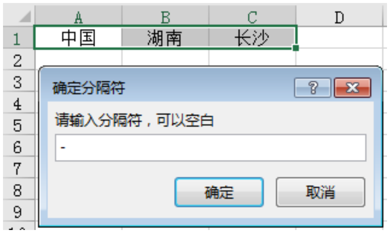


图 13–37　图片库菜单的预览效果　　图 13–38　当鼠标指针指向库时显示功能预览与提示

（10）假设 A1:C1 区域分别有“中国”、“湖南”、“长沙”三个字符串，选择 A1:C1 区域后选择菜单“开始”→“合并单元格”→“合并居中”，程序将弹出如图 13-39 所示的提示框。

（11）在“确定分隔符”对话框中可以随意定义分隔符，也可以保持默认的“-”作为分隔符。假设选择“-”作为分隔符，单击“确定”按钮后，A1:C1 区域将合并为如图 13-40 所示效果。



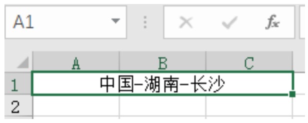


图 13–39　指定分隔符　　图 13–40　合并效果

（12）选择合并后的 A1 单元格，再依次选择菜单“开始”→“合并单元格”→“取消合并”，在弹出的“确定分隔符”对话框中保持默认的“-”作为分隔符，然后单击“确定”按钮，将合并后的 A1:C1 区域转换为合并前的状态，即 A1、B1 和 C1 三个单元格分别存放“中国”、“湖南”、“长沙”三个字符串。

【知识补充】

（1）库是 Excel 2007 开始应用在功能区中的一项新的技术，通过它可以实现需要执行的功能的效果预览，让用户在执行过程之前就能看到最终效果，从一定程度上实现了菜单的智能化。

（2）如果修改代码，还可以实现让库显示多行多列，并且任意调整每行每列的显示高度与宽度，也可以随意指定图片的路径。

（3）在将硬盘中的图片绑定到库时，不能直接将路径赋值给参数，而是需要使用 LoadPicture 将图片路径转换成图片对象，否则库无法识别。

本例案例文件请参考：..\ 第 13 章\ 13-16 创建图片库.xlsm

13.4 使用模板

功能区设计比传统的菜单与工具栏更强大，同时也更复杂，在编写代码时会有不小的困难，所以最好的办法是设计一个或者多个模板来简化工作。

本节详述模板的重要性和设计方法。

13.4.1 模板的重要性

模板就是预先做好的样本，可以重复使用，提供参考作用，简化工作量。

功能区设计是不可以录制的，所以对用户的要求较高，必须懂得各种控件的语法，而事实上 VBA 用户往往不是专业程序员，没有精力也没有必要去记忆这些语法。在此前提下，模板就显得格外重要。

功能区模板是指预先做好的包含各种功能区组件的 XML 文件，在后续设计功能区时直接调用模板中的代码并稍加修改即可。

事实上，Custom UI Editor 软件就自带 5 个模板。

13.4.2 模板的使用方法

Custom UI Editor 软件自带 5 个模板，保存在以下路径：

64 位系统：

```
C:\Program Files (x86)\Custom UI Editor\Samples
```

32 位系统：

```
C:\Program Files\Custom UI Editor\Samples
```

读者可以根据自己的实际情况打开文件路径，查看软件自带的模板。

在以上路径下的 5 个文件都是 XML 格式，用记事本打开即可看到其中的源代码。

模板文件默认采用 Excel 2010 格式，即包含“<customUI>”的根元素中使用了“2009/07”，所以代码不能在 Excel 2007 中正常使用，读者将其修改为“2006/01”即可。

当然，以上仅说明模板文件的保存路径，在实际调用模板时选择菜单“Insert”→“Sample XML”下的文件名称即可，不用担心文件存放在何处。图 13-41 展示了如何调用模板文件中的代码。

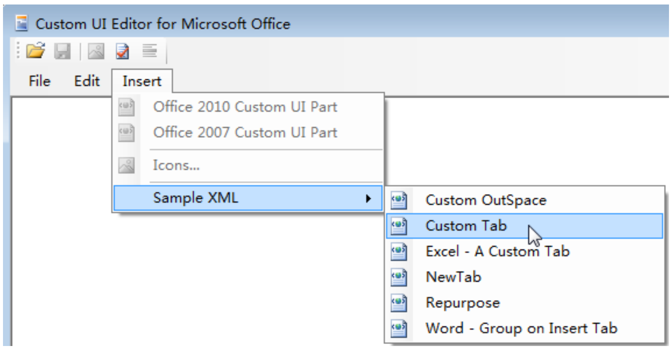


图 13-41 调用模板文件中的代码

13.4.3 制作两个模板

制作模板可以是全功能型模板，也可以是有针对性的单一型模板。

全功能型模板可以将常用的功能区控件都放进去，使用时调出此代码并删除不需要的部分即

可，其优势是一个模板可解决绝大部分问题。

单一型模板是根据需求制作多个模板，每一个模板应对一种需求。例如弹出式菜单、复选框、对话框启动器、切换按钮、图片库……其优势是调用速度快，特别是对于无法读懂代码的新手来说。

接下来展示制作两个模板的过程，从而加深读者对模板的理解。

1．全功能型模板

【模板要求】模板需包含组、标签、命令按钮、切换按钮、弹出式菜单、复选框、对话框启动器、下拉列表和编辑框等常用控件。

【操作步骤】

（1）从 Windows 的菜单“开始”中打开“记事本”软件，从而创建一个空白的 txt 文件；

（2）在文件中录入以下代码，代码用于创建包括组、标签、命令按钮、切换按钮、弹出式菜单、复选框、对话框启动器、下拉列表和编辑框等常用控件。

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon>
    <tabs>
      <tab id="新选项卡" label="新选项卡" insertAfterMso="TabHome">
       <group id="group1" label="One">
        <labelControl id="label1" label="标签"/>
        <button id="customButton1" label="命令按钮 1" screentip="提示"
supertip="详细提示"  onAction="Macro1" imageMso="Club" />
        <toggleButton id="toggleButton1" label="切换按钮" visible="true"
enabled="true"  onAction="zero"  imageMso="AnimationAudio" size="normal"
screentip="零值切换" supertip="按下时显示零值，弹起时不显示零值" keytip="L"/>
        <separator id="分隔条 1" />
        <menu id="One" label="弹出式菜单" screentip="Excelbbx"
supertip="http://excelbbx.net" size="large"  imageMso="AppointmentColorDialog" >
          <button id="customButton2" label="命令按钮 2"  screentip="提示"
supertip="详细提示"  onAction="Macro3" imageMso="AccessListCustom" />
          <menuSeparator id="菜单分隔条" />
          <button id="customButton3" label="命令按钮 3"  screentip="提示"
supertip="详细提示"  onAction="Macro4" imageMso="AddressBook" />
        </menu>
        <labelControl id="label2" label="单击时切换"/>
          <checkBox id="checkBox1" label="复选框 1"  onAction="Macro5"/>
          <checkBox id="checkBox2" label="复选框 2"  onAction="Macro6"/>
          <separator id="分隔条 2" />
         <button  id="customButton4"  label="命令按钮 4"  screentip="提示"  supertip="详细提示"
onAction="Macro7" size="large" imageMso="AccessListEvents"
/>
        <dialogBoxLauncher><button id="dialogBox1" label="对话框启动器"
screentip="提示" supertip="详细提示" onAction="Macro8" /> </dialogBoxLauncher>
        </group>
       <group id="group2" label="Two" >
        <dropDown id="Style" showLabel="true" label="匹配方式"onAction="
dropDownChange" >
         <item id="xlWhole" label="xlWhole" imageMso="WatchWindow" />
         <item id="xlPart" label="xlPart" imageMso="ZoomPrintPreviewExcel" />
```

```
                </dropDown>
                <editBox   id="FindTxt"   label="  查  找  内  容  "   imageMso="ZoomPrintPreviewExcel"
sizeString="999999999999"  maxLength="30"  visible="true"  showLabel="true"     onChange="editBoxChange"
keytip="R" />
              </group>
            </tab>
          </tabs>
        </ribbon>
      </customUI>
```

在以上代码中分别采用“标签”、“分隔条 1”、“命令按钮 1”、“命令按钮 2” 等对控件的 label 参数赋值，从而便于识别当前代码所创建的控件的名称。

（4）使用【Ctrl+S】组合键保存文件，在“另存为”对话框中将文件名称设置为“多功能样本.xml”，同时从“编码”列表中选择“Unicode”，如图 13-42 所示。

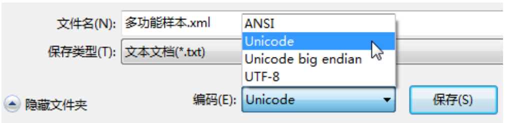


图 13-42　保存文件

（4）选择保存文件的路径，64 位系统：

```
C:\Program Files (x86)\Custom UI Editor\Samples
```

32 位系统：

```
C:\Program Files\Custom UI Editor\Samples
```

然后单击“保存”按钮。

（4）打开 Custom UI Editor 软件，选择菜单“Insert”→“Sample XML”，将看到新的模板名称“多功能样本”，单击模板名称可以将代码插入到当前窗口中。

在实际使用时，可以在导入模板中的代码后根据需要删除多余的控件代码，并对剩下的代码改下菜单名称即可,也可以直接复制需要的部分,例如组、按钮、复选框等。

图 13-43 是本例模板文件的最终预览效果。

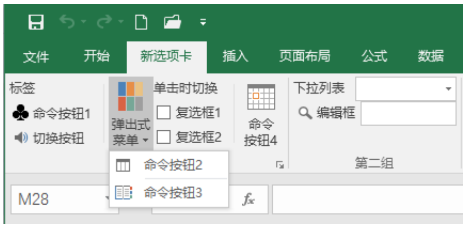


图 13-43　模板文件包含的功能区控件预览效果

本例案例文件请参考：..\ 第 13 章\多功能样本.xml

2. 单一型模板

【模板要求】模板主要展示组的编写方式，例如在“开始”选项卡中的组、插入到中间和放在最末端、新选项卡中的组，以及“视图”选项卡中的组。通过此模板可以对以后创建组提供便利。

【操作步骤】

（1）从 Windows 的菜单“开始”中打开“记事本”软件，从而创建一个空白的 TXT 文件。

（2）在文件中录入以下代码，代码用于创建位于不同位置的四个组。

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
 <ribbon startFromScratch="false">
  <tabs>
     <!-- 以下代码表示在开始选项卡中创建两个组，首尾各一个-->
    <tab idMso="TabHome" >
     <group id="Group1" label="第一组" insertBeforeMso="GroupFont">
     </group>
     <group id="Group2" label="第二组" >
     </group>
    </tab>
     <!-- 以下代码表示在新建选项卡中创建一个组-->
   <tab id="CustomTab" label="新选项卡" insertAfterMso="TabHome">
     <group id="Group3" label="第三组">
     </group>
   </tab>
     <!-- 以下代码表示在视图选项卡的最前面创建一个组-->
   <tab idMso="TabView" >
     <group id="Group4" label="第四组" insertBeforeMso="GroupWorkbookViews">
     </group>
   </tab>
  </tabs>
 </ribbon>
</customUI>
```

代码中以“<!—”开头、“-->”结尾的注释，在执行代码时会自动跳过，不会产生语法问题。使用注释后更利于后续的编写工作与代码维护。

（3）使用【Ctrl+S】组合键保存文件，在“另存为”对话框中将文件名称设置为“组模板.xml”，同时从“编码”列表中选择“Unicode”。

（4）选择保存文件的路径为 Custom UI Editor 软件的模板目录 Samples，然后单击“保存”按钮。

（5）打开 Custom UI Editor 软件，选择菜单“Insert”→“Sample XML”，将看到新的模板名称“组模板”，单击模板名称可以将代码插入到当前窗口中。

以下四张图片是以上模板文件的最终效果预览（由于 Excel 2016 中不会显示空白的组，因此在 Excel 中截图给读者查看的组中添加了按钮等元素，以便在 Excel 2016 中正常显示组）。

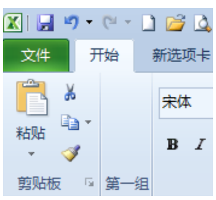


图 13-44　第一组



图 13-45　第二组

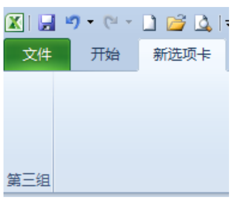


图 13-46　第三组

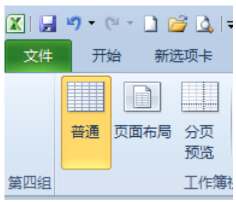


图 13-47　第四组

本例模板中的代码仅针对组 group 提供样本，读者可以借助此思路再分别对切换按钮、下拉列表等控件制作模板。

本例案例文件请参考：..\ 第 13 章\组模板.xml

13.5 课后思考

1．在编写定制功能区的 XML 代码时，什么情况下需要区分字母大小写？什么情况不需要区分大小写？

2．在“审阅”选项卡末端添加一个标签，标签中显示打开文件的当日是星期几。

3．利用 XML 代码可以隐藏所有功能区选项卡吗？

4．在选项卡中添加一个命令按钮，将 C 盘中的名为“A.jpg”的图片作为该按钮的图标。

5．idMso 与 id 的区别是什么？

第 14 章　开发通用插件

插件的存在价值在于简化操作，将 VBA 代码转换成插件能有效提高代码的应用效率。

插件是 Sub 过程的工具版，可以将插件理解为一个工具箱或者安装包，所以在使用方面会更方便。

本章主要介绍插件的类别 、特点和开发步骤。

本章要点

◆ 插件的分类

◆ 漫谈加载宏

◆ 制作工作表批量重命名插件

14.1　插件的分类

Excel 允许使用多种插件，针对不同的目的应选用不同的插件格式。

在开发插件之前有必要了解插件的种类和各自的特点，在此基础上才能有的放矢。

14.1.1　什么是插件

Excel 的插件是指用于强化 Excel 功能的外置安装文件，它不属于 Excel 应用程序，但安装到 Excel 后可以绑定到 Excel 界面之中，实现 Excel 原本无法实现的一些功能。

开发 Excel 的插件有较多方法，最方便的是直接使用 VBA 编写，然后将工作簿另存为加载宏。加载宏工作簿就是一个插件。

14.1.2　插件的分类方式

插件包含加载项和加载宏两大类。

加载项通常是 DLL 或 OCX 格式，是将 VBA 代码封装后的插件。加载项较 VBA 具有更高的安全性，用户只能使用插件而不能查看代码。

加载宏是直接由 Excel 工作簿转换而成的具有特殊功能的文件，它制作简单，使用方便，但是安全性不高，用户可以随意查看其中的源代码。

加载宏包含 xla 和 xlam 两种文件格式，其中 xla 格式是 Excel 97 至 Excel 2003 版本对应的加载宏文件格式；xlam 格式则是 Excel 2007 后新增的加载宏格式，它是 xla 格式的增强版，可储存的数据更多，而且在储存相同数据的前提下，xlam 格式的文件体积更小。

在兼容性方面，xla 格式的加载宏强于 xlam 格式的加载宏，可以同时在 Excel 2000、Excel 2003、Excel 2007、Excel 2010、Excel 2013 和 Excel 2016 中使用，而 xlam 格式的加载宏则只能在 Excel 2007、Excel 2010、Excel 2013 和 Excel 2016 中使用。

读者可以根据需求选用适合自己的文件格式。

14.1.3 开发插件和编写普通代码的分别

在本章之前所编写的代码通常都是针对代码所在工作簿执行操作的。

而有时某些代码是通用的，在多个工作簿中都需要调用同一个 Sub 过程，那么这部分 Sub 过程就比较适宜存放在加载宏工作簿中，从而在用户调用过程中提供便利。

简而言之，普通的 Sub 过程仅用于解决某个问题，而插件则用于解决某类问题。

目的不同，编写代码的思路也就不同。

普通过程仅为解决当前问题，那么编写代码时针对性较强。例如有明确的区域地址、工作表名称或者文件路径，那么在代码中可以直接采用硬编码，包括 Range("A1:G400")、Worksheets("生产表")、“C:\工作文件”等。

而在编写插件时需要解决一类问题，考虑问题的思路会完全不同。例如需要开发一个对区域按颜色求和的插件，在开发插件时求和区域属于未知数，没有明确的地址编码，所以编写代码只能采用变量代替区域地址。可采用 Application.Inputbox 方法弹出一个对话框让用户选择区域，而不能采用 Range("A1:G400")这类代码强行指定区域地址。

与之类似的问题还有文件路径、工作簿名称、图片名称等，一切都需要考虑通用性、兼容性，所以开发插件和开发普通的 Sub 过程的思路有较大的区别。

此外开发插件会涉及版本问题，例如插件的终端用户有可能使用 Excel 2003，也有可能使用 Excel 2010、Excel 2016，因此在制作菜单时需要同时兼顾多个版本的用户。

再如解决重复值问题，由于实现需求的方法较多，在编写普通的过程时直接使用本机所支持的方式即可；在开发插件时，开发者往往并不是终端用户，所以有必要选择既适合 Excel 2003 用户的方法，也适合 Excel 2007、Excel 2010 和 Excel 2016 用户的方法。

14.2 漫谈加载宏

加载宏是最简单的 Excel 插件，开发方法简单，使用方法也较简单。本节先介绍加载宏的特性、使用方法，下一节演示制作加载宏的具体步骤。

14.2.1 加载宏工作簿的特点

加载宏是一个工具，同时也是一个工作簿，只不过它是具有工具特性的工作簿。

加载宏工作簿有以下特点。

- 加载宏的 IsAddin 属性为 true。
- 加载宏不属于 Workbooks 成员，在使用“Workbooks.count”计数时会忽略加载宏。
- 加载宏的窗口是隐藏的，并且无法通过功能区中任何功能按钮操纵加载宏工作簿。
- 加载宏安装一次即可在所有工作簿中调用。
- 无法通过【Alt+F8】组合键方式调用加载宏中的任何程序。
- 加载宏后台定义的函数可以在任何打开的工作簿中使用，加载宏创建的菜单也将出现在所有工作簿中，可以通过菜单访问加载宏中的任何程序。
- 加载宏的结构无法在前端修改，如增减工作表、拆分窗口、设置工作表背景、删除工作表中的数据等，从而保护加载宏工作簿数据及程序的正常运行。

将带有宏的普通工作簿另存为加载宏的优势主要有以下三个。

（1）加载宏中的代码可以在所有工作簿中调用，不需要将代码复制到目标工作簿中，从而提升了工作效率。

（2）所有代码存放在加载宏中，而不是日报表、库存表之类的工作簿，管理代码更方便。

（3）在运行安装后的加载宏时不会出现关于宏的安全性方面的警告信息。

将普通工作簿转换成加载宏的方法是将文件另存，在“另存为”对话框中将“保存类型”设置为加载宏即可。图 14-1 展示了在“另存为”对话框中将文件保存为加载宏。

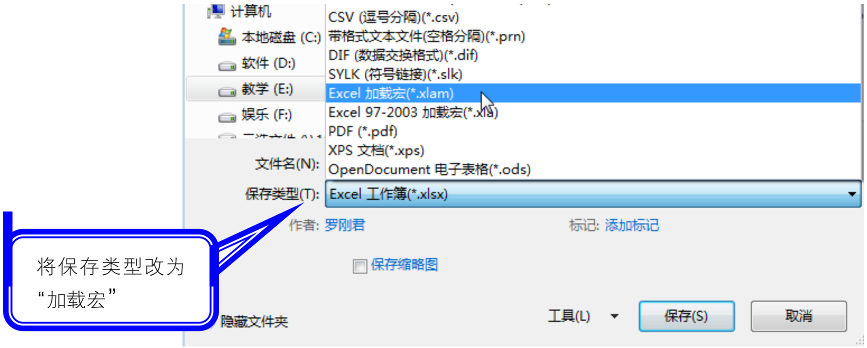


图 14–1　将文件保存为“加载宏”

如果需要将加载宏还原为 xlsm 或者 xls 格式，最快捷的方法是用代码实现。

目前加载宏包括两种格式，Excel 97 至 Excel 2016 通用的“*.xla”格式，Excel 2007、Excel 2010、Excel 2013 和 Excel 2016 专用的“*.xlam”格式。如果确定加载宏的目标用户会使用 Excel 2003，那么加载宏格式应采用“*.xla”，否则应尽量采用“*.xlam”格式，后者才能使用功能区。

14.2.2　加载宏管理器

Excel 提供了加载宏管理器，它用于显示当前的加载宏列表，并通过复选框是否打勾来呈现加载宏的可用状态，也可以通过“浏览”按钮添加新的加载宏。

打开加载宏管理器的快捷键是【Alt+T+I】。

默认状态下在加载宏管理器中有若干个内置的加载宏，例如“标签打印向导”、“欧元向导”、“规划求解加载项”等。如果安装了自定义的加载宏，那么将按拼音顺序一并罗列出来，如图 14-2 所示。

如果某文件已打勾，那么表示该加载宏处于可用状态。在图 14-3 中已经勾选了“规划求解加载项”，因此在“数据”功能区中可以看到“规划求解”的菜单，如图 14-3 所示。

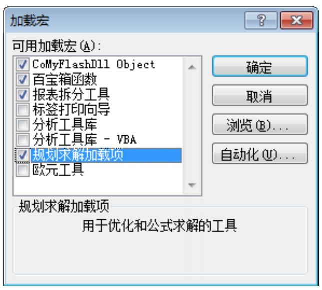


图 14–2　加载宏管理器

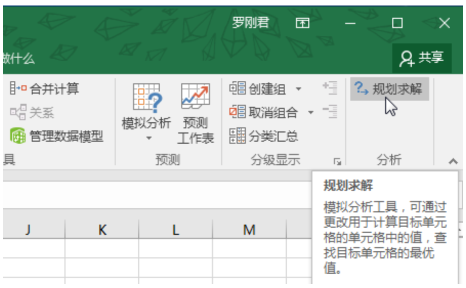


图 14–3　规划求解工具

14.2.3 加载宏的使用方法

加载宏是一个工作簿，同时它也是一个工具，所以在使用加载宏之前需要安装。虽然双击打开加载宏也可以调用加载宏中的功能，但是这种调用方式仅对本次操作有效，下次打开 Excel 软件就无法再次调用加载宏中的功能。为了一劳永逸，使得以后可以随时调用加载宏，需要安装加载宏文件。

安装加载宏有两种方法，包括加载宏管理器法和自启动路径法。

现以“C:\工具\另存图片.xlam”为例，展示安装加载宏的基本步骤。

1. 加载宏管理器

使用加载宏管理器安装加载宏的步骤如下。

（1）在 Excel 工作表界面按下【Alt+T+I】组合键打开“加载宏”对话框。

（2）单击“浏览”按钮，在“C:\工具\”路径下，选择文件“另存图片.xlam”并单击“确定”按钮返回“加载宏”对话框，在对话框中将罗列出新装的加载项，如图 14-4 所示。

（3）任意插入一张图片，并选中图片，在“格式”选项卡中将看到名为“另存为”的命令按钮，如图 14-5 所示，此命令按钮即属于“C:\工具\另存图片.xlam”中的工具菜单。

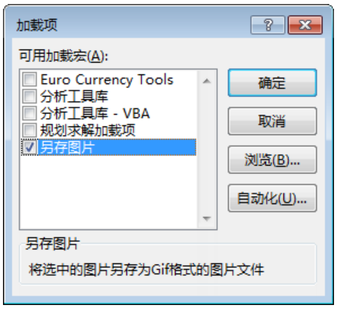


图 14–4 加载宏管理器

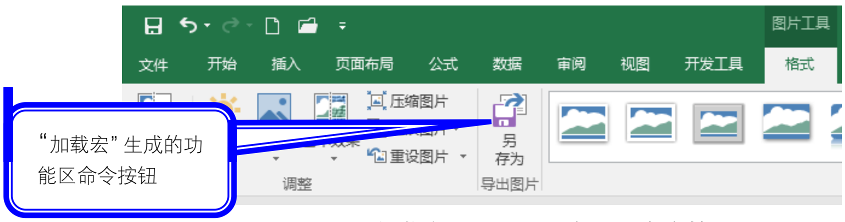


图 14–5 加载宏生成的“另存为”命令按钮

本例案例文件请参考：..\ 第 14 章\ 另存图片.xlam

2. 自启动路径

安装加载宏的目的就是让文件随着 Excel 软件启动而自动启动，从而方便调用，所以如果将加载宏文件放在自启动路径下就相当于安装加载宏了。

自启动路径较多，最便于记忆的是 XLSTART 文件夹，此文件夹路径如下：

C:\Program Files\Microsoft Office\Office11\XLSTART——Excel 2003 版的路径

C:\Program Files\Microsoft Office\Office12\XLSTART——Excel 2007 版的路径

C:\Program Files\Microsoft Office\Office14\XLSTART——Excel 2010 版的路径

C:\Program Files\Microsoft Office\Office15\XLSTART——Excel 2013 版的路径

如果读者使用了 64 位操作系统，那么“Program Files”应修改为“Program Files (64)”。

Excel 2016 的自启动路径有较大的改动，其路径如下：

```
C:\Users\Administrator\AppData\Roaming\Microsoft\Excel\XLSTART
```

其中 Administrator 是用户名称，用不同用户名称登录电脑，这个路径就会不同。

只要将加载宏工作簿保存在以上自启动路径中，那么打开 Excel 软件时会同步打开加载宏文件，从而可以随时调用加载宏中的过程。

14.2.4　加载宏的便利性

Excel“信任中心”的宏设置对加载宏无效，不管设置为“禁用所有宏，并且不通知”、“禁用所有宏，并发出通知”，还是“所用无数字签署的所有宏”，加载宏中的代码都可以顺利运行。

基于此规则，将宏代码制作成加载宏并发给用户使用时，不需要叮嘱“必须修改宏设置”或者“请启用宏”，从而避免代码被默认设置所阻挡，导致无法运行。

14.3　制作工作表批量重命名插件

制作加载宏的重点不在于将普通工作簿转成加载宏，而在于加载宏中的 Sub 过程和菜单。本节以开发“批量重命名”加载宏为例，展示开发加载宏的详细步骤。

14.3.1　开发通用插件的基本步骤

加载宏属于通用插件，专为解决某类工作需求而存在，而非满足临时性的工作需求，所以代码要确保具有通用性、兼容性，能在不同的环境中使用。

当然，为了确保用户调用插件中的便捷性，通常需要为过程设置快捷键或者菜单，还需要添加帮助窗体，告知用户使用方法。

开发通用插件的基本步骤如下。

（1）罗列插件需求。

（2）设计窗体，包括主程序界面和帮助信息。

（3）编写窗体代码。

（4）创建菜单窗体的菜单。如果更偏重于兼容性则采用传统菜单，否则采用功能区。

（5）将文件保存为加载宏。

（6）安装并测试插件。

14.3.2　罗列插件需求

开发插件前极有必要罗列插件的功能需求，这和写作前先拟定提纲一样重要。

工作表批量重命名插件需有以下功能：

（1）对工作表命名的数据源应有两种产生方式：一是手动录入，二是来自区域引用。

（2）允许选择命名的范围，而非总是对所有工作表重命名。

（3）在重命名时应提供三种命名方式，包括“插入到原名称前”、“插入到原名称后”和“替换原名称”。

（4）提供启动插件的快捷键。

14.3.3　设计插件窗体

插件需要两个窗体，包括“工作表批量重命名”主程序和用于帮助的说明窗体。

1．主程序界面设计

创建主程序界面步骤如下。

（1）新建一个工作簿，按【Alt+F11】组合键进入 VBE 界面。

（2）选择菜单“插入”→“用户窗体”，从而创建一个空白的窗体。

（3）如果没有显示“属性”对话框则按【F4】打开“属性”对话框，然后在对话框中将窗体的“名称”属性修改为“ReName”，将“Caption”属性修改为“工作表批量重命名”。

（4）单击工具箱中的“多页”控件，然后在窗体中按下鼠标并拖放，从而添加一个默认包含两页的“多页”控件。

（5）将“多页”控件的第一页的“名称”属性修改为“手动指定”，将第二页修改为“引用区域”，此时窗体效果如图 14-6 所示。

（6）在第一页中添加五个标签、三个文本框、三个选项按钮、一个命令按钮和两个框架，排列方式如图 14-7 所示。其中第三个选项按钮的“Value”属性设置为 True，表示默认按“替换原名称”方式重命名。

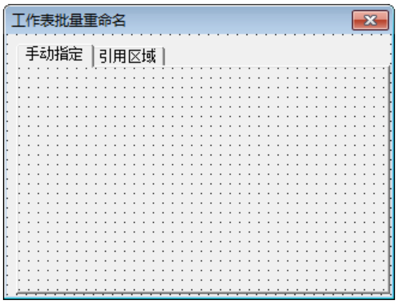


图 14–6　创建窗体及包含两页的多页控件

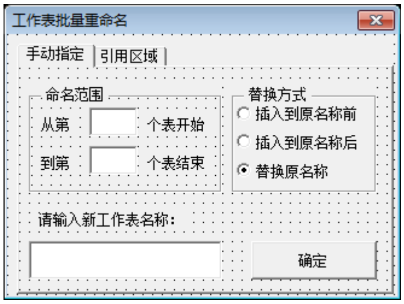


图 14–7　第一页的排列方式

（7）为了方便区分和识别，分别将从上到下的三个文本框分别命名为“开始 A”、“结束 A”和“新表名称 A”。

（8）将三个选项按钮的“名称”属性分别命名为“插入到原名称前 A”、“插入到原名称后 A”和“替换原名称 A”；将“Caption”属性分别修改为“插入到原名称前”、“插入到原名称后”和“替换原名称”。

（9）将命令按钮的“名称”属性修改为“确定 A”，将“Caption”属性修改为“确定”。

（10）进入“多页”控件的第二页，添加五个标签、三个文本框、三个选项按钮、一个命令按钮和两个框架，排列方式如图 14-8 所示。其中第三个选项按钮的“Value”属性设置为 True，表示默认按“替换原名称”方式重命名。

（11）将从上到下的三个文本框分别命名为“开始 B”、“结束 B”和“新表名称 B”。

（12）将三个选项按钮的“名称”属性分别修改为“插入到原名称前 B”、“插入到原名称后 B”和“替换原名称 B”，将“Caption”属性分别修改为“插入到原名称前”、“插入到原名称后”和“替换原名称”。

对文本框、选项按钮和命令按钮命名的作用是区分控件处于“多页控件”的第一页还是第二页。其中包含“A”者，表示处于第一页；包含“B”者，表示处于第二页。

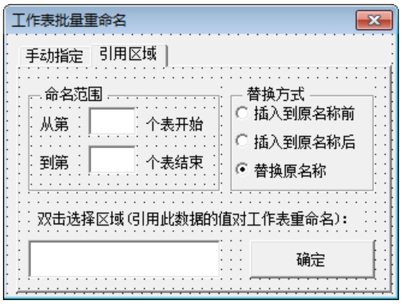


图 14–8　第二页的排列方式

2．帮助窗体

帮助窗体仅用于说明插件的功能，所以比较简单，通过动画或者标签提供文字描述即可，不需要代码。

本例利用 Flash 动画和文字标签说明插件的功能和用法，所以需要预先准备一个 Flash 动画。

设计帮助窗体的步骤如下。

（1）选择菜单“插入”→“用户窗体”，从而创建一个空白的窗体。

（2）在属性窗口中将窗体的“名称”属性修改为“Help”，将“Caption”属性修改为“重命名工具箱说明”。

（3）在工具箱的空白区单击右键，从快捷菜单中选择“附加控件”，在弹出的“附加控件”窗口中勾选“Shockwave Flash Object”，然后单击“确定”按钮。

（4）单击工具箱中的“ShockwaveFlash”控件，在窗体中拖放，生成一个“ShockwaveFlash”控件。

（5）在属性窗口中将“ShockwaveFlash”控件的“Movie”属性修改为 Flash 动画的完整路径，并且将“EmbedMovie”属性设置为 True，表示将 Flash 动画嵌入到工作簿中。

（6）在窗体中添加两个标签，在标签中说明插件的用法。

（7）为了收集插件用户的反馈信息与建议，在窗体说明中需要留下作者的个人联系方式。窗体的最终效果如图 14-9 所示。

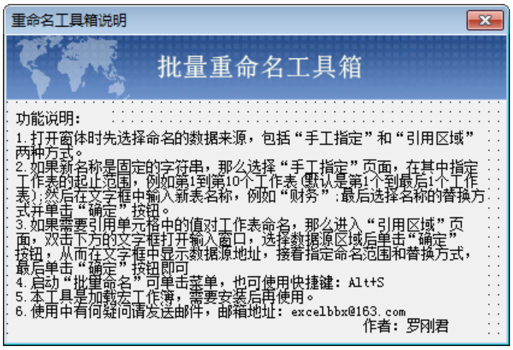


图 14–9　帮助窗体最终效果

14.3.4　编写代码

由于帮助窗体不需要代码，所以仅需编写主程序窗体的代码。主程序窗体中需要四段代码，源代码如下。

```
Private Sub UserForm_Activate()          '激活窗体时执行（放置位置：模块中）
    开始 A = 1                            '文本框“开始 A”的默认值为 1
    结束 A = Sheets.Count                 '文本框“结束 A”的默认值为工作表数量
    开始 B = 1                            '文本框“开始 B”的默认值为 1
    结束 B = Sheets.Count                 '文本框“结束 B”的默认值为工作表数量
End Sub
```

以上代码在显示窗体时自动执行，功能是为 4 个代表工作表范围的文本框指定初始值。

```
Private Sub 确定 A_Click()
 '如果未指定数值的起止范围则提示用户，然后结束过程
  If Not IsNumeric(开始 A.Value) Or Not IsNumeric(结束 A.Value) Then MsgBox "请输入数字！": Exit Sub
  '如果起始数大于结束数则提示用户，然后结束过程
  If 开始 A.Value * 1 >= 结束 A.Value * 1 Then MsgBox "终止数必须大于起始数！": Exit Sub
  '如果终止数大于工作表总数量则提示用户，然后结束过程（要注意“Exit Sub”和冒号在同一行中）
  If 结束 A.Value > Sheets.Count Then MsgBox "终止数不能大于工作表数量：" & Sheets.Count: Exit Sub
  '如果未指定新的工作表名称则提示用户，然后结束过程
  If Len(新名称 A) = 0 Then MsgBox "请输入新工作表名！": Exit Sub
  '声明两个变量，前者用于循环语句的变量，后者用于计数器，表示已命名的工作表数量
  Dim ShtItem As Integer, i As Integer
  For ShtItem = 开始 A.Value To 结束 A.Value        '遍历需要重命名的每个工作表
    If 插入到原名称前 A Then                          '如果选项按钮“插入到原名称前 A”处于选中状态
        '将变量 ShtItem 所代表的工作表命名为：新名称连接原名称
        Sheets(ShtItem).Name = 新名称 A.Text & Sheets(ShtItem).Name
    ElseIf 插入到原名称后 A Then                      '如果选项按钮“插入到原名称后 A”处于选中状态
        '将变量 ShtItem 所代表的工作表命名为：原名称连接新名称
        Sheets(ShtItem).Name = Sheets(ShtItem).Name & 新名称 A.Text
    ElseIf 替换原名称 A Then                          '如果选项按钮“替换原名称 A”处于选中状态
        '将变量 ShtItem 所代表的工作表命名为：新名称连接自然数序号
        i = i + 1
        Sheets(ShtItem).Name = 新名称 A.Text & (i):
    End If
  Next
End Sub
```

以上代码在单击第一页的“确定”按钮时执行，功能是根据用户的设置对工作表批量命名。该过程首先执行了 4 次判断，用于防错。即用户设置不正确时通过这 4 句代码提醒用户，且自动中断过程，避免在执行后面的代码时产生错误。一个完善的插件必定含有多个必要的条件语句，用于预防不规范的操作。

当确保设置正确后，可以通过 For Next 循环语句和 If Then 条件语句对工作表批量命名。需要注意的是当替换方式设置为“替换原名称”时不能使用代码“Sheets(ShtItem).Name = 新名称 A.Text”，因为在同一工作簿中不允许存在多个同名的工作表。本例的解决办法是在后面添加升序的自然数序号加以区分。

```
'双击名为“新名称 B”的文本框时执行
Private Sub 新名称 B_DblClick(ByVal Cancel As MSForms.ReturnBoolean) '放置位置：模块中
    Dim adds As String                   '声明一个 String 型的变量，用于保存选区地址
    On Error Resume Next                 '当代码出错时，继续运行下一句代码
    '将当前选区的地址赋值给变量 adds
    adds = ActiveWindow.RangeSelection.Address
    Me.Hide                              '隐藏窗体，便于选择区域
    Dim Rng As Range                     '声明一个 Range 型变量，用于保存用户选择的区域
```

```
        '弹出一个输入框让用户选择命名的数据来源
        Set Rng = Application.InputBox("请选择单元格区域：", "确定区域", adds, , , , , 8)
        新名称 B = Rng.Address              '将用户选择的区域的地址赋值给名为“新名称 B”的文本框
        Me.Show                             '显示窗体
    End Sub
```

以上代码在双击名为“新名称 B”的文本框时执行，其功能是在双击“新名称 B”文本框时弹出一个对话框，让用户指定数据源地址，在对工作表重命名时将会调用此区域中的数据。

```
    '单击第二页的"确定"按钮时执行
    Private Sub 确定 B_Click()                          '放置位置：模块中
        On Error GoTo err                               '当错误时执行 err 标签后的语句
    '如果未指定数值的起止范围则提示用户，然后结束过程
     If Not IsNumeric(开始 B.Value) Or Not IsNumeric(结束 B.Value) Then MsgBox "请输入数字！": Exit Sub
    '如果起始数大于结束数则提示用户，然后结束过程
     If 开始 B.Value * 1 >= 结束 B.Value * 1 Then MsgBox "终止数必须大于起始数！": Exit Sub
    '如果终止数大于工作表总数量则提示用户，然后结束过程
     If 结束 B.Value > Sheets.Count Then MsgBox "终止数不能大于工作表数量：" & Sheets.Count: Exit Sub
     '如果未指定新的工作表名称则提示用户，然后结束过程
      If Len(新名称 B) = 0 Then MsgBox "请双击文本框选择单元格区域！": Exit Sub
     '声明三个变量
      Dim ShtItem As Integer, Item As Integer, NewName As String
      For ShtItem = 开始 B.Value To 结束 B.Value        '遍历需要重命名的每个工作表
           Item = Item + 1                              '累加计数器
          '将指定的区域中第 Item 个单元格的值赋予变量 NewName
           NewName = Range(新名称 B)(Item).Text
           If 插入到原名称前 B Then                      '如果选项按钮“插入到原名称前 B”处于选中状态
                '将变量 ShtItem 所代表的工作表命名为：新名称连接原名称
                 Sheets(ShtItem).Name = NewName & Sheets(ShtItem).Name
            ElseIf 插入到原名称后 B Then                 '如果选项按钮“插入到原名称后 B”处于选中状态
                '将变量 ShtItem 所代表的工作表命名为：原名称连接新名称
                Sheets(ShtItem).Name = Sheets(ShtItem).Name & NewName
            ElseIf 替换原名称 B Then                     '如果选项按钮“替换原名称 B”处于选中状态
                '将变量 NewName 的值赋予工作表名称
                Sheets(ShtItem).Name = NewName
            End If
        Next
        Exit Sub                                        '结束过程
    err:                                                '设置一个标签，过程执行出错时将执行此处的代码
        MsgBox "您的选区中存在重复值或者空白区域，或者新名称与原工作表名重复,程序无法继续！", 64, "提
示"
    End Sub
```

以上代码在单击第二页的“确定”按钮时执行，其功能是根据用户的设置对工作表批量命名。命名前的判断语句和过程“确定 A_Click”一致，命名时的命名方式也一致，差异仅在于命名的数据源不同，前者来自文本框“新名称 A”中的字符串，后者来自文本框指定区域中的每一个单元格。

由于在用户指定的区域中可能存在重复值或者空白单元格，所以在这个过程中需要使用防错语句“On Error GoTo err”。

设计好窗体后，将文件保存为“批量重命名.xlam”。

14.3.5 创建菜单与设置快捷键

为“批量重命名.xlam”创建功能区菜单，需要使用 Custom UI Editor 软件，操作步骤如下。

（1）打开 Custom UI Editor 软件，从 Custom UI Editor 软件中打开“批量重命名.xlam”。

（2）选择菜单“Insert”→“Office 2007 Custom UI Part”，从而插入一个 customUI.xml 文件。

（3）在右边的代码窗口中录入以下代码。

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon>
    <tabs>
      <tab id="NewTab" label="重命名" insertAfterMso="TabHome">
       <group id="group1" label="批量重命名">
        <button id="customButton1" label="批量重命名" screentip="对工作表批量命名" supertip="对活动工作簿中指定的工作表批量重命名，包含在前方插入新名称、在后方插入新名称，用新名称替换原名称。" onAction="重命名 1" imageMso="AddOrRemoveAttendees" size="large"/>
        <button id="customButton2" label="帮助" screentip="帮助" supertip="功能说明"  onAction="帮助1" imageMso="TentativeAcceptInvitation" size="large"/>
       </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

（4）单击“保存”按钮，然后关闭 Custom UI Editor 软件。

（5）双击打开“批量重命名.xlam”，然后按【Alt+F11】组合键进入 VBE 界面。

（6）选择菜单“插入”→“模块”，然后录入以下代码。

```
'打开工作簿时自动运行
Sub Auto_open()                          '放置位置：模块中
    Application.OnKey "%s", "重命名 2"   '对过程“重命名 2”指定快捷键为“Alt+S”
  End Sub
Sub 重命名 2()
  ReName.Show 0                          '显示窗体 ReName
End Sub
Sub 重命名(control As IRibbonControl)
  ReName.Show 0                          '显示窗体 ReName
End Sub
Sub 帮助(control As IRibbonControl)
  Help.Show                              '显示窗体 Help
End Sub
```

（7）使用【Ctrl+S】组合键保存代码，然后关闭工作簿。

完成以上步骤之后，插件已经设计完成，可以在任意电脑中安装此插件。

本例案例文件请参考：..\ 第 14 章\批量重命名.xlam

14.3.6 安装并测试功能

每当插件开发完成后，一定需要进行多次测试，包括测试本身的功能是否满足需求，以及插件是否适应不同环境，环境包括 Excel 2007、Excel 2010、Excel 2013 及 Excel 2016 等。

1. 安装

（1）在工作表界面按【Alt+T+I】组合键打开“加载宏”对话框。

（2）单击“浏览”按钮打开“浏览”对话框，在对话框中找到“批量重命名.xlam”的存放路径，然后选中“批量重命名.xlam”并单击“确定”按钮返回“加载项”对话框，此时对话框的加载宏列表中会新增一项“批量重命名”，效果如图 14-10 所示。

（3）关闭“加载项”对话框返回工作表界面，此可以看到名为“重命名”的选项卡，在该选项卡中有“批量重命名”和“帮助”两个子菜单，效果如图 14-11 所示。

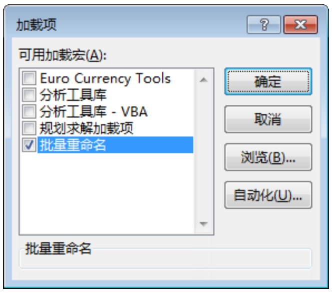


图 14-10　安装加载项

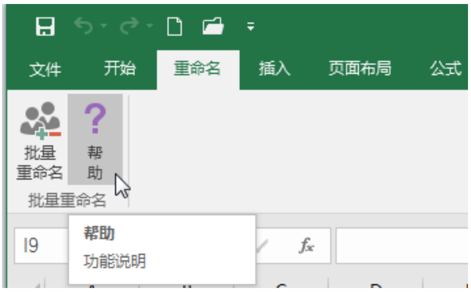


图 14-11　安装插件后的界面

2．测试

安装插件后，可以打开需要重命名的文件测试插件的功能。

假设工作簿中有 7 个工作表，按图 14-12 所示的方式排列，对它批量重命名步骤如下。

（1）使用【Alt+S】组合键打开主程序。

（2）将命名范围设置为第 1 到第 6 个表，替换方式设置为“插入到原名称前”，将新的工作表名称设置为“10 月”，然后单击“确定”按钮，执行结果如图 14-13 所示。

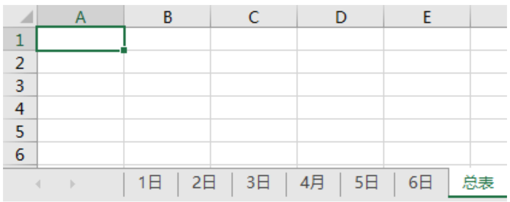


图 14-12　工作表布局

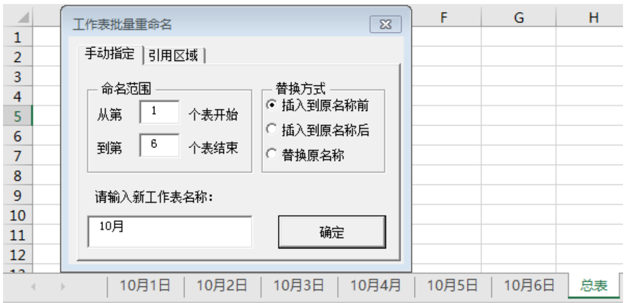


图 14-13　新名称插入到原名称前

由于设置时采用的命名范围是第 1 到第 6 个表，因此第 7 个表“总表”并没有命名。

（3）将新的工作表名称修改为“生产表”，将替换方式修改为“插入到原名称后”，然后单击“确定”按钮，重命名结果如图 14-14 所示。

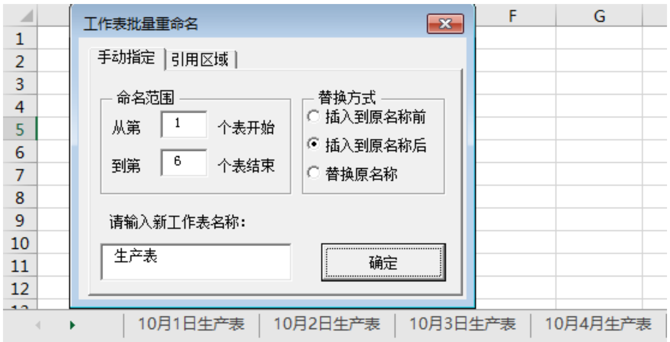


图 14-14　新名称插入到原名称之后

（4）将工作表范围设置为第 1 到第 7 个表，将新的工作表名称修改为“Sheet”，将替换方式设置为“替换原名称”，然后单击“确定”按钮，重命名结果如图 14-15 所示。

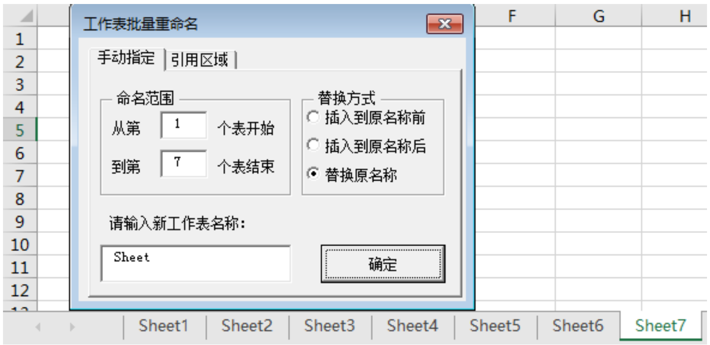


图 14-15　替换原名称

（5）返回工作表界面，在 A1:A7 区域分别录入“星期一”到“星期日”7 个字符串，然后按【Alt+S】组合键打开重命名工具的主窗体界面。

（6）进入“引用区域”页面，双击下方的文本框，然后选择 A1:A7 区域，将在“确定区域”对话框中产生选区地址，如图 14-16 所示。

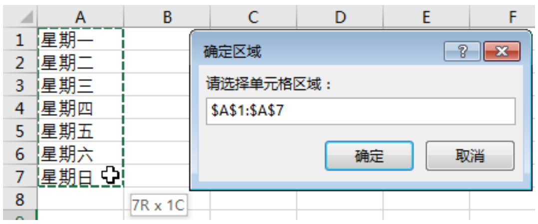


图 14-16　选择数据源

（7）单击“确定区域”对话框中的“确定”按钮返回主窗体，再将替换方式修改为“替换原名称”，然后单击“确定”按钮，命名结果如图 14-17 所示。

经过以上在 Excel 2016 环境中的测试，可以确定插件已经能够正常工作。

笔者也在 Excel 2010 和 Excel 2007 中测试通过，由于结果完全一致，不再一一演示。读者可以使用不同的工作簿，采用不同的字符串进行测试。

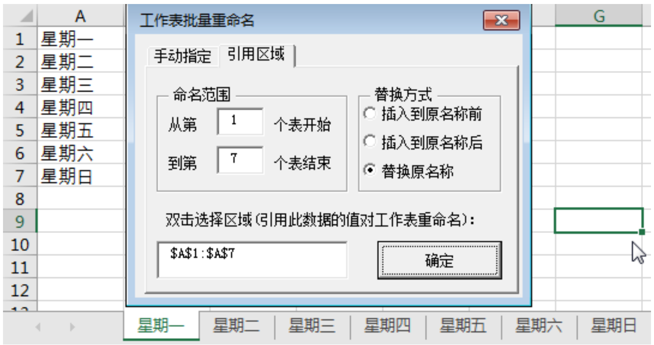


图 14-17　使用“引用区域”的值批量命名工作表

14.4　课后思考

1．双击加载宏工作簿，为什么看不到工作表?
2．如何删除已经安装的自定义加载宏文件?
3．如何将加载宏文件还原为普通文件？例如 xlsm 或者 xls 文件。
4．对名为“NewSheet”的 Sub 过程指定组合键【Alt+Shift+D】。
5．修改本章的加载宏，使其在重命名时仅针对工作表，自动跳过宏表或者图表。

第 15 章　让 VBA 代码也能撤销

VBA 可以让代码自动化执行，从而提升工作效率。

不过 VBA 的一个最明显的缺陷是执行代码后不能撤销，破坏了【Ctrl+Z】组合键的功能。如果将插件交给客户，客户在使用过程中覆盖了重要数据，那么会影响其对软件的好评率。所以插件应尽量提供撤销功能，哪怕只能撤销最后一步。

本章要点

◆ 突破撤销限制

◆ 可撤销的简体转繁体插件

15.1　突破撤销限制

Excel VBA 有很多优点，对工作的帮助较大，不过不允许撤销这个最大的缺点往往掩盖了它的所有优点，致使部分用户放弃使用 VBA。所幸通过 Application.OnUndo 方法可让用户返回上一步，在一定程度上突破了这种限制。

15.1.1　VBA 命令的撤销限制

VBA 默认禁止执行代码后返回上一步状态，不过它同时又提供了 Application. OnUndo 方法让用户撤销最后一次操作。

Application.OnUndo 方法事实上并不是撤销，而是允许开发者编写一个过程，关联到“撤销”按钮，用户在单击该按钮或者按下【Ctrl+Z】键时可以调用此过程。

因此，编写程序时重点在于提供一个配套的过程用于返回上一步的状态。例如过程 A 和过程 B，过程 A 用于记录工作表的当前状态，过程 B 则用于恢复数据，当用户按下【Ctrl+Z】时会调用过程 B 恢复过程 A 备份的数据。

由于 Application.OnUndo 方法的这种特性，将引申出以下四个问题。

其一，撤销操作其实就是将备份的数据复制到当前表，因此必须要有一个可存放数据的空白工作表，而加载宏中的工作表正是一个空白工作表，可以将活动工作表的数据备份到其中，在需要时再还原。基于此原理，在设计加载宏时可以让代码允许撤销，而普通工作簿中的 Sub 过程在执行后却不宜撤销，因为找不到方便存取的空白工作表。

其二，在复制、粘贴数据后，再把数据复制回去可以确保数据不错位，完全还原。但是把工作表中所有图片备份后再粘贴回去却会错位，因此为了避免用户抱怨撤销功能有 Bug，应直接告诉用户撤销功能仅针对数据，不针对图形对象。

其三，由于现阶段 xls 格式和 xlsm、xlsx 格式混合使用，在恢复数据时需要考虑工作表的行数问题，避免恢复失败。例如加载宏采用了 xlam 格式，其工作表行数为 1048576，而用户的活动工作表是 xls 格式，只有 65536 行，将 1048576 行的工作表中整表复制到只有 65536 行的工

作表注定会失败，因此在复制前需要做判断，然后采用不同的复制方式。

其四，某些操作是没有办法撤销的，例如删除磁盘中的文件、删除工作表、关闭打开的工作簿等操作，因此在对 VBA 的 Sub 过程设计撤销功能时也应该只针对单元格中写入数据这类操作，其他操作则不提供撤销功能。况且 Excel 自身的撤销操作也是有选择性的，对于新建工作表、删除工作表等操作也不允许撤销。

15.1.2　设计可以撤销的 Sub 过程的思路与步骤

Application.OnUndo 方法的语法如下。

```
Application.OnUndo(Text, Procedure)
```

参数 Text 用于描述当前需要撤销的操作，可以随意指定，该文本将显示在快速访问工具栏中的撤销列表中；参数 Procedure 表示关联到“撤销”按钮的过程名称，当用户单击“撤销”按钮或者按下【Ctrl+Z】组合键时可以调用此过程。

利用 Application. OnUndo 方法开发一个可以撤销的过程的常规思路如下。

首先，声明一个 Worksheet 型的公共变量（用于指定需要恢复数据的工作表）以及一个 String 型的公共变量（用于指定需要恢复数据的区域地址）。

其次，编写一个备份数据的过程和撤销的过程，分别命名为“备份”、“撤销”（使用其他名称也可以）。其中“备份”过程带有两个参数，分别为需要备份的工作表对象和区域地址。

最后，在需要使用撤销功能的过程中调用“备份”和“撤销”两个过程，其中调用“备份”过程的代码放在过程中修改单元格的代码上方，即先备份后修改；调用“撤销”过程的代码放在过程的末尾，即“Exit Sub”或者“End Sub”之前。

图 15-1 是撤销功能的效果展示，当执行过程“标示重复值”后鼠标指针指向“撤销”按钮时会提示“撤销【标示重复值】”。

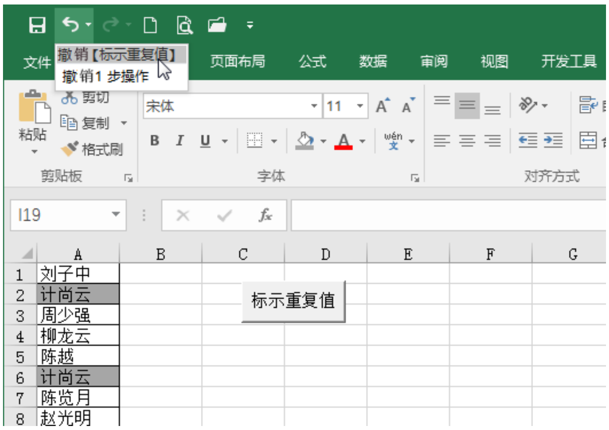


图 15–1　执行过程后在“撤销”铵钮处显示指定的文字

注意：备份数据时最好的办法原本是将整个工作表备份，这样能完整地备份数据和格式，但是这种备份方式有严重的隐患，它会占用较大的内存，当电脑的内存小时就可能会提示内存不足。因此在备份数据时应该只备份有数据的区域，尽管此方式也不完善，会忽略列宽、隐藏等格式，但是基于两害相权取其轻的原则，仍然应该采用此方式备份数据。

15.2 设计可撤销的插件

插件属于工具类，而不再是一个单纯的工作簿。插件的终端用户和开发者不是同一个人，因此更应当注重撤销功能。本节将展示拥有“按颜色汇总”和“取消合并单元格并填充数据”两个功能的插件开发过程，此插件中的两个功能都支持撤销，但它们的撤销过程又稍有区别，通过具体代表性的案例演示，读者可以对撤销代码有更全面的认识。

15.2.1 编写插件

在设计可撤销的插件前，可先按以下步骤开发一个普通的加载宏，当测试通过后再加入撤销功能。

（1）打开 Excel 2016，然后将当前空白工作簿保存为加载宏文件，名称为“Excel 精灵（无撤销）.xlam”。

（2）打开 Custom UI Editor 软件，从 Custom UI Editor 软件中打开“Excel 精灵（无撤销）.xlam”。

（3）选择菜单“Insert”→“Office 2007 Custom UI Part”从而插入一个 customUI.xml 文件。

（4）在右边的代码窗口中录入以下代码。

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon>
    <tabs>
      <tab id="NewTab" label="Excel 精灵" insertAfterMso="TabHome">
       <group id="group1" label="可撤销插件展示">
         <button id="customButton1" label="按颜色汇总" screentip="按颜色汇总" supertip="对选中的区域按单元格背景色汇总，汇总包含计数与求和。" onAction="按颜色汇总" imageMso="SlideMasterChartPlaceholderInsert" size="large"/>
         <button id="customButton2" label="取消合并单元格并填充数据" screentip="取消合并单元格填充数据" supertip="取消所有合并单元格，并且对取消合并后产生的空白单元格填充数据" onAction="取消合并单元格并填充数据" imageMso= "TableSharePointListsRefreshList" size="large"/>
      </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

（5）保存代码，然后关闭 Custom UI Editor 软件。

（6）在 Excel 中打开“Excel 精灵（无撤销）.xlam”，按下【Alt+F11】组合键进入 VBE 界面。

（7）选择菜单“插入”→“模块”，然后录入以下两段代码。

第一段：

```
Sub 取消合并单元格并填充数据(control As IRibbonControl)       '放置位置：模块中
  Dim Rng As Range                                            '声明一个 Range 型的对象变量
  Application.FindFormat.Clear                                '清除查找格式
  Application.FindFormat.MergeCells = True                    '重新指定查找格式，即查找合并单元格
  With Cells                                                  '引用活动工作表的所有单元格
   '按设定的格式查找
    Set Rng = .Find(what:="", LookIn:=xlFormulas, LookAt:=xlPart, SearchFormat:=True)
    If Rng Is Nothing Then MsgBox "没有合并单元格": Exit Sub '如果未找到则结束过程
    Application.ScreenUpdating = False"                       '关闭屏幕更新，加快执行速度
    Do  '开始循环
      With Rng.MergeArea                                      '引用找到的目标单元格的合并区域
```

```
            .UnMerge                          '取消合并
            '让原本合并区域中的每一个单元格都等于该区域第一个单元格的值
            .Cells = .Cells(1).Value
          End With
          Set Rng = .Find(what:="", after:=Rng, SearchFormat:=True) '查找下一个
          If Rng Is Nothing Then              '如果找不到（表示取消完成）
             Application.ScreenUpdating = True '恢复屏幕更新
             Exit Sub                         '结束过程
          End If
       Loop
    End With
End Sub
```

第二段：

```
Sub 按颜色汇总(control As IRibbonControl)   '放置位置：模块中
    On Error Resume Next                    '当程序出错时继续执行下一步
    Dim cell As Range, i As Integer, j As Integer, Rng As Range, Rng2 As Range, temp As Long, SumRng As
Range                                       '声明变量
    '如果选择的对象不是单元格，则结束过程
    If TypeName(Selection) <> "Range" Then MsgBox "请选择待合计区域！ ": Exit Sub
    If WorksheetFunction.CountA(Selection) <= 2 Then MsgBox "请选择一个稍大范围的非空区域再测试 ", 64:
Exit Sub                                    '如果选区的数字小于等于 2 则结束过程"
    If Selection.Rows.Count = Rows.Count Or Selection.Columns.Count = Columns.Count Then MsgBox "不
要选择整行、整列", 64: Exit Sub              '整行整列选择也结束过程
  '将已用区域与选区的交集赋予变量 Rng2
    Set Rng2 = Intersect(ActiveSheet.UsedRange, Selection)
    '利用对话框让用户选择存放区域
    Set Rng = Application.InputBox("请选择存放区域， 一个单元格即可！ ", "确认目标地址",
Selection.Rows(Selection.Rows.Count + 1).cells(1).Address, , , , , 8)
    Set Rng = Rng(1)                        '将 Rng 重置为 Rng 区域的第一个单元格
  '如果有错误（单击了“取消”按钮），则结束过程
    If err <> 0 Then MsgBox "请填写正确的汇总区域地址": err.Clear: Exit Sub
    If Rng.Parent.Name = ActiveSheet.Name Then
      '如果存放结果的区域与待合计的区域重叠，则结束过程
      If Not Intersect(Rng, Selection) Is Nothing Then MsgBox "不能让数据区与汇总结果重叠" & Chr(10) & "请
重新选择结果存放区域": Exit Sub
    End If
    Application.Calculation = xlCalculationManual    '设置为手动计算，以提升程序的执行效率
    Application.ScreenUpdating = False               '关闭屏幕更新，以提升程序扫行效率
    Rng.Resize(Selection.Rows.Count / 2, 3).Clear    '清空存放结果的区域（由于选区中一定会多个单
  '元格拥有相同颜色，因此存放结果的区域一定小于当前选区的一半）
    Rng.Resize(1, 3) = Array("颜色", "数量", "合计")  '在结果存放区域的第一行写上标题
    err.Clear                                        '清除错误
    For Each cell In Rng2        '遍历 Rng2 区域，即当前选区与活动工作表的已用区域的交集
      On Error Resume Next                           '当程序出错时继续执行下一步
      If IsNumeric(cell.Text) Then                    '如果单元格中是数字
       '计算单元格的颜色在存放结果的区域中第二行的排位序号
        j = WorksheetFunction.Match(cell.Interior.Color, Rng.Resize(i + 1, 1), 0)
        If err <> 0 Then         '如果有错误（当在该区域中没有找到与 cell 单元格颜色一致的编号时，
  '程序会出错，而非像 Match 函数那样返回一个错误值，这和在工作表中使用函数是不一样的）
            err.Clear            '清除错误
```

```
                i = i + 1                    '累加变量，该变量等于选区中颜色的个数
                On Error GoTo err            '如果有错误，则执行 err 标签处的语句（此处如果有错误，通常
'表示写入数据失败，此时程序无法继续执行，只能结束过程）
                Rng.Offset(i, 0) = cell.Interior.Color          '在存放结果区域的第一列中写入颜色编号
                Rng.Offset(i, 0).Interior.Color = cell.Interior.Color '再将颜色设置为与 cell 的颜色一致
                Rng.Offset(i, 2) = cell.Value                   '将 cell 的值引用到结果区域的第三列中
                Rng.Offset(i, 1) = 1                            '在结果区域的第二列中写入数值 1，表示次数为 1
            Else                                                '如果没有错误（表示区域中已经有该颜色编号）
                Rng.Offset(j - 1, 2) = Rng.Offset(i, 2).Value + cell.Value '将 cell 的值累加到结果单元格中
                Rng.Offset(j - 1, 1) = Rng.Offset(j - 1, 1) + 1 '将次数也累加到结果单元格中
            End If
        End If
    Next cell
    Rng.Offset(1, 0).Resize(i, 1).Value = ""                   '将第 1 列清空（仅保留颜色）
    Rng.Resize(i + 1, 3).Borders.LineStyle = xlContinuous       '对整个区域添加边框
    Application.Calculation = xlCalculationAutomatic           '恢复自运计算
    Application.ScreenUpdating = True                          '恢复屏幕更新
    Exit Sub                                                   '退出程序，避免执行 err 标签后面的语句
err:                                                           '设置一个标签
    MsgBox "请重新选择结果存放区域。"
    Application.ScreenUpdating = True                           '恢复屏幕更新
End Sub
```

（8）按下【Ctrl+S】组合键，保存代码。

（9）按本书 14.3.6 节的方法安装加载宏，然后在 Excel 的功能区中会产生如图 15-2 所示的新菜单，当鼠标指针指向菜单时会生成文字说明。

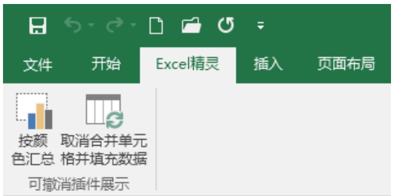
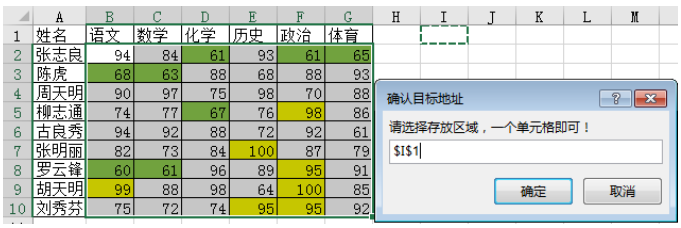


图 15-2　插件的菜单

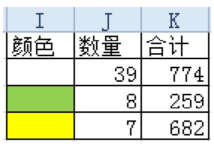
安装好后需要测试菜单功能。

（10）打开“测试数据.xlsx”的 Sheet1，选择数据区域 B2:G10，然后选择菜单“按颜色求和”，在弹出对话框后选择 I1 单元格，然后单击“确定”按钮，此时 Excel 会对选区中的数据按颜色汇总。图 15-3 和图 15-4 分别代表指定结果存放区域和汇总结果。



姓名	语文	数学	化学	历史	政治	体育
张志良	94	84	61	93	61	65
陈虎	68	63	88	68	88	93
周天明	90	97	75	98	70	88
柳志通	74	77	67	76	98	86
古良秀	94	92	88	72	92	61
张明丽	82	73	84	100	87	79
罗云锋	60	61	96	89	95	91
胡天明	99	88	98	64	100	85
刘秀芬	75	72	74	95	95	92

图 15-3　指定结果存放区域



颜色	数量	合计
	39	774
	8	259
	7	682

图 15-4　汇总结果

（11）打开 Sheet2，选择菜单“取消合并单元格并填充数据”，工具中的所有合并单元格将取消合并，同时会在取消合并产生的空白单元格内填充数据。在图 15-5 中左图是取消合并前的

效果，右图是取消合并后的效果。



图 15–5　取消合并单元格前后效果对比

经过以上测试表明插件的功能正常，一个普通的插件开发完成。

本例案例文件请参考：..\ 第 15 章\Excel 精灵（无撤销）.xlam　+ 测试数据.xlsx

15.2.2　为插件添加撤消功能

在上一小节中设计了一个不可撤销的插件，包含“按颜色汇总”和“取消合并单元格并填充数据”两个功能，本节对它们添加撤销功能，具体步骤如下。

（1）复制“Excel 精灵（无撤销）.xlam”，将其重命名为“Excel 精灵.xlam”，然后打开 Excel 2016，进入 VBE 窗口，并双击“Excel 精灵.xlam”文件的模块 1 进入代码窗口。

（2）在过程“取消合并单元格并填充数据”上方添加以下代码，用于声明公共变量。

```
'声明两个公共变量，其中 Targetsht 代表需要备份数据的工作表
'TargetRng 代表需要备份数据的区域的地址
Public Targetsht As Worksheet, TargetRng As String
```

（3）在过程“按颜色汇总”下方添加以下两个过程，分别用于备份数据和撤销数据。

```
'备份数据的过程，其中参数 Sht 代表要备份的工作表对象，rngAddress 则代表要备份的区域地址
Sub 备份(sht As Worksheet, rngAddress As String)
  ThisWorkbook.Sheets(1).Cells.Clear '先删除 ThisWorkbook 第一个工作表的所有单元格的值
 '然后将 sht 工作表的 rngAddress 区域复制到 ThisWorkbook 第一个工作表的 A1
 sht.Range(rngAddress).Copy ThisWorkbook.Sheets(1).Range(rngAddress)
End Sub
Sub 撤销()
  '将 ThisWorkbook 的第一个工作表的 TargetRng 区域的值恢复到 TargetSht 表中
  '要注意两个工作表的区域是对应的，都采用 TargetRng
  ThisWorkbook.Sheets(1).Range(TargetRng).Copy TargetSht.Range(TargetRng)
End Sub
```

（4）在过程“取消合并单元格并填充数据”中的 Application.ScreenUpdating = False 下方插入以下代码，其功能是在修改单元格之前备份数据，并对变量 TargetSht 和 TargetRng 赋值，在后续撤销时将要用到这两个变量，它们分别代表数据所在工作表和地址。

```
 Set TargetSht = ActiveSheet      '对公共变量赋值，在执行撤销时会用到 TargetSht
  '对公共变量赋值，在执行备份和撤销时会用到 TargetRng
  TargetRng = TargetSht.UsedRange.Address
  Call 备份(TargetSht, TargetRng) '调用“备份”过程
```

（5）在过程“取消合并单元格并填充数据”中的 Application.ScreenUpdating =True 上方插入以下代码，其功能是在修改单元格之后为“撤销”按钮指定 Sub 过程及要显示的文字。

```
'设置在单击“撤销”按钮时要显示的文字和关联的过程
Application.OnUndo "撤销 【取消并填充合并单元格】", "撤销"
```

（6）在过程“按颜色汇总”中的 Application.ScreenUpdating ＝ False 下方插入以下代码，其功能是在修改单元格之前备份数据，并对变量 TargetSht 和 TargetRng 赋值。

```
'对公共变量 TargetSht 赋值为 Rng 的父对象，即存放结果的工作表。
'在执行撤销时会用到变量 TargetSht
Set TargetSht = Rng.Parent
'对公共变量 TargetRng 赋值为 TargetSht 的已用区域的地址。
'在执行备份和撤销时会用到变量 TargetRng
TargetRng = TargetSht.UsedRange.Address
 Call 备份(TargetSht, TargetRng) '调用“备份”过程
```

（7）在过程“按颜色汇总”中的 Application.ScreenUpdating = True 下方插入以下代码，其功能是在修改单元格之前备份数据，并对变量 TargetSht 和 TargetRng 赋值。

```
Application.OnUndo "撤销 【按颜色合计】", "撤销" '设置“撤销”按钮上要显示的文字和关联的过程
'再次记录已用区域的地址（因为执行过程后已用区域可能会变大，为了避免出现恢复数据后无法覆盖原来的数据，导致让用户认为撤销失败的情况，此处必须重新记录已用区域的地址）
TargetRng = TargetSht.UsedRange.Address
```

（8）按下【Ctrl+S】组合键保存代码，然后重启 Excel。

（9）再次打开“测试数据.xlsm”，进入 Sheet1，选择 B2:G10 区域后再单击功能区的“Excel 精灵”→“按颜色汇总”，在弹出输入框后选择 I1 单元格，然后单击“确定”按钮，在 I1:K4 区域中生产生合计结果。

（10）将鼠标指针移向“撤销”按钮，可以看到“取消【按颜色合计】”的提示，如图 15-6 所示。单击按钮后工作表中 I1:K4 区域的汇总结果会自动消失，表明撤销成功。

如果执行“按颜色汇总”，将结果存放在其他工作表，那么撤销时也会针对该工作表，而非仅限于活动工作表。

（11）进入“测试数据.xlsm”的 Sheet2，选择功能区的“Excel 精灵”→“取消合并单元格并填充数据”，工作表中所有合并单元格都会取消合并，同时填充所有数据。此时将鼠标指针移向“撤销”按钮可以看到新的提示“撤销【取消并填充合并单元格】”，效果如图 15-7 所示。

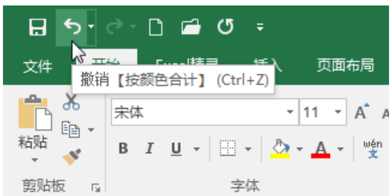


图 15-6 “撤销”按钮提示

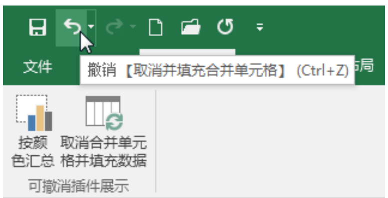


图 15-7 “撤销”按钮提示

（12）单击“撤销”按钮或者按下【Ctrl+Z】组合键，此时活动工作表会恢复到取消合并单元格前的状态。

本例案例文件请参考：..\ 第 15 章\ Excel 精灵.xlam

【代码分析】

（1）当工程较大时，有可能在一个工程中有多个模块，那么每个模块中的过程都需要撤消，因此变量 TargetSht 和 TargetRng 要确保所有模块都能调用才行，在声明这两个变量时要用 Public，而非 Dim。

（2）备份数据工作应该在修改单元格之前完成，否则无法恢复数据。所以在使用“Call 备份(Targetsht, TargetRng)”时要判断当前过程中哪句代码会修改单元格的值，找准位置后再插入代码执行备份。

备份前需要对变量 TargetSht 和 TargetRng 赋值。在绝大多数情况下，统一使用以下方式赋值即可。

```
Set TargetSht = ActiveSheet
TargetRng = ActiveSheet.UsedRange.Address
```

因为工作中绝大多数的 Sub 过程都用于修改活动工作表中的单元格，因此 TargetSht 和 TargetRng 都采用活动工作表即可。

对少数需要修改其他工作表数据的 Sub 过程，则采用其他工作表来替代 ActiveSheet。本例中“按颜色汇总”的汇总结果有可能放在活动工作表中，也可能放在其他工作表中，具体是哪一个工作表得由变量 Rng 决定，即用户选择了哪个工作表的单元格，那么需要备份、撤销的对象就是该工作表。因此在“按颜色汇总”过程中采用 Rng.Parent 替换 ActiveSheet。

本书特意用两个不同的案例来教学，实际上是为了提升知识的全面性以及工具的实用性。

（3）在备份数据时，数据存放在加载宏工作簿中。ThisWorkbook 代表加载工作簿，不能将数据保存在 Activeworkbook 中。

备份时先清除上一次保存的值，然后用以下代码备份数据。

```
Sht.Range(rngAddress).Copy ThisWorkbook.Sheets(1).Range(rngAddress)
```

其中 Sht 代表要备份的工作表，它在多数情况下等于 Activesheet，在少数情况下表示其他工作表，具体由 Sub 过程中的赋值方式而定。针对不同的需求，会采用不同的赋值方式。

ThisWorkbook.Sheets(1).Range(rngAddress)表示存放备份数据的地方，即加载宏工作簿的第一个工作表的 rngAddress 区域。其中 rngAddress 区域等于要备份的工作表的 Usedrange 的地址。

（4）rngAddress 变量会使用两次，第一次是在修改单元格数据之前备份数据时使用，第二次是在修改单元格数据之后恢复数据时使用。对于备份时要使用的 rngAddress，统一采用修改单元格之前的 TargetSht.UsedRange.Address 即可；至于在恢复数据时要使用的 rngAddress，则要区别对待，如果在修改数据时不会增大 UsedRange 区域，那么调用修改数据前的 TargetSht.UsedRange.Address，否则必须调用修改数据后的 TargetSht.UsedRange.Address，即在修改数据之后再次对变量 rngAddress 赋值。

在本例过程“按颜色汇总”中对 rngAddress 赋值了两次，这是因为在过程中按颜色汇总后有可能会增大 TargetSht.UsedRange。如果在恢复数据时仅恢复到增大 Targetsht.UsedRange 前的区域，就会无法完整覆盖数据，导致撤销失败。

通过图 15-8 可以明白对 rngAddress 赋值两次的必要性。如果只赋值一次，那么在备份时只备份了 A1:G10，在撤销时只覆盖 A1:G10 区域，因此 I1:K4 区域的值仍然存在，代表撤销失败。如果赋值两次，在第二次赋值时对 rngAddress 赋值为“A1:K10”，那么在撤销时就会覆盖 A1:K10 区域，从而让 I1:K4 区域的值消失，撤销成功。

（5）设计插件时需要考虑哪些 Sub 过程需要撤销，哪些 Sub 过程不需要撤销。通常采用一个原则：有可能会破坏单元格中数据的操作都需要撤销，其他过程不需要撤销。

以“合并工作表”的功能为例，如果合并结果存放在活动工作表中，那么它就有可能破坏数据，需要撤销；如果合并结果存放在一个新建的工作表中，那么不需要撤销，因为此操作并不具有任何破坏性，如果合并错了则手动删除新建的工作表即可。

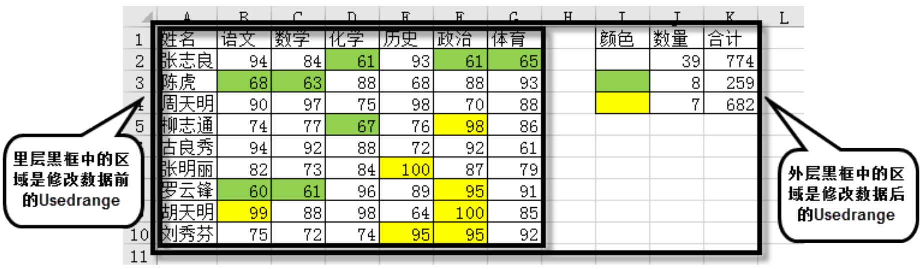


图 15-8　UsedRange 修改数据前后的变化图

再如“拆分工作簿”，将一个工作簿中每个工作表单独保存为一个文件，存放在指定的路径下，此操作也没有任何破坏性，原来的数据都是安全的，因此即使操作错了也不需要撤销，手动删除文件即可。还有“重命名工作表”、“插入图片”等操作也是一样的。

当然，原则上对“拆分工作簿”、“合并工表”、“重命名工作表”、“新建文件夹”等操作执行撤销也是可以办到的，只是看有没有必要性而已。

（6）以“拆分工作簿”为例，如果一定要对修改单元格之外的操作也实现撤销，而且要与前面的案例中的撤销并存，那么可以按以下方式操作。

首先复制“Excel 精灵.xlam”，并命名为“Excel 精灵(含拆分工作簿).xlam”，用 Custom UI Editor 软件打开它，将原本的 XML 代码按如下方式修改，也就是添加一个名为“拆分工作簿”的按钮。

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon>
    <tabs>
      <tab id="NewTab" label="Excel 精灵" insertAfterMso="TabHome">
       <group id="group1" label="可撤销插件展示">
        <button id="customButton1" label="按颜色汇总" screentip="按颜色汇总" supertip="对选中的区域
按单元格背景色汇总，汇总包含计数与求和。"  onAction="按颜色汇总"
imageMso="SlideMasterChartPlaceholderInsert" size="large"/>
        <button id="customButton2" label="取消合并单元格并填充数据" screentip="取消合并单元格填充
数据" supertip="取消所有合并单元格，并且对取消合并后产生的空白单元格填充数据"  onAction="取消合并单元
格并填充数据" imageMso= "TableSharePointListsRefreshList" size="large"/>
         <button id="customButton3" label="拆分工作簿" screentip="拆分工作簿" supertip="将活动工作
簿中的每个工作表另存为一个单独的工作簿"  onAction="拆分工作簿" imageMso="AdpDiagramArrangeTables"
size="large"/>
      </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

然后双击打开文件“Excel 精灵(含拆分工作簿).xlam”，在模块顶端添加以下代码，用于声明两个公共变量。

```
'声明两个变量，第一个变量用于记录拆分工作簿时生成的文件名称
'第二个变量代表“撤销”的方式
Public FileArr() As String, style As String
```

接着在模块最下方添加如下代码，即拆分工作簿的过程。

```
Sub 拆分工作簿(control As IRibbonControl)                  '放置位置：模块中
  Dim PathSht As String                                    '声明变量，用于存放路径
  With Application.FileDialog(msoFileDialogFolderPicker)   '让用户选路径
   '如果用户选择了路径，就记录下路径，否则结束过程
```

```
        If .Show Then PathSht = .SelectedItems(1) Else Exit Sub
    End With
    '如果路径右方一个字符不是“\”则追加一个“\”
    PathSht = PathSht & IIf(Right(PathSht, 1) = "\", "", "\")
    Dim acwb As Workbook                    '声明一个变量，用它来代表活动工作簿
    Set acwkb = ActiveWorkbook              '对变量赋值
    Application.ScreenUpdating = False      '关闭屏幕更新
    For Item = 1 To acwkb.Sheets.Count      '遍历所有工作表
        Sheets(Item).Copy '将第 Item 个工作表复制到新工作簿中
        '将新工作簿另存到指定的路径下（用工作表名作为工作簿名称）
        ActiveWorkbook.SaveAs PathSht & acwkb.Sheets(Item).Name & ".xlsx", xlOpenXMLWorkbook
        ActiveWorkbook.Close False          '关闭新工作簿
    Next Item
    Application.ScreenUpdating = True       '恢复屏幕更新
End Sub
```

同时将“撤销”过程按以下方式修改：

```
Sub 撤销()'此过程在用户单击“撤销”按钮或者按【Ctrl+Z】时执行，撤销时要执行何种操作由变量 style
'决定，通常情况下恢复数据即可，特殊情况下由变量 style 的值而定
'如果没有对 style 参数赋值（即针对通常的情况。“按颜色汇总”和“取消合并单元格并填充数据”
'都属于通常的情况）
    If style = "" Then
        '将 ThisWorkbook 的第一个工作表的 TargetRng 区域的值恢复到 TargetSht 表中
        '要注意两个工作表的区域是对应的，都采用 TargetRng
        ThisWorkbook.Sheets(1).Range(TargetRng).Copy TargetSht.Range(TargetRng)
        '如果变量 style 赋值为“拆分工作簿”
    ElseIf style = "拆分工作簿" Then
        On error Resume Next                '当程序出错时，继续执行下一句代码
        Dim item As Integer                 '声明变量，用于循环
        For item = 1 To UBound(FileArr)     '遍历数组的每个元素
            Kill FileArr(item)              '逐个删除文件
        Next item
        style = ""                          '将变量恢复为默认值，避免影响其他 Sub 过程的撤销操作
        Erase FileArr                       '释放数组变量的值，避免影响下一轮的工作簿拆分
    End If
End Sub
```

最后按【Ctrl+S】组合键保存代码，并关闭 Excel 即可。

测试代码的方法和前面一样，按本书 14.3.6节的方法安装“Excel 精灵(含拆分工作簿).xlam”，然后打开“测试数据.xlsx”文件，单击菜单“拆分工作簿”，在弹出的“浏览”对话框中选择“D:\拆分结果”，单击“确定”按钮后，程序会将活动工作簿中的 3 个工作表拆分成 3 个工作簿，保存在“D:\拆分结果”路径中，效果如图 15-9 所示。

如果此时按下【Ctrl+Z】组合键，那么“D:\拆分结果”路径中的 3 个文件会自动消失，表示撤销成功。

本例案例文件请参考：..\ 第 15 章\ Excel 精灵(含拆分工作簿).xlam + 测试数据.xlsx

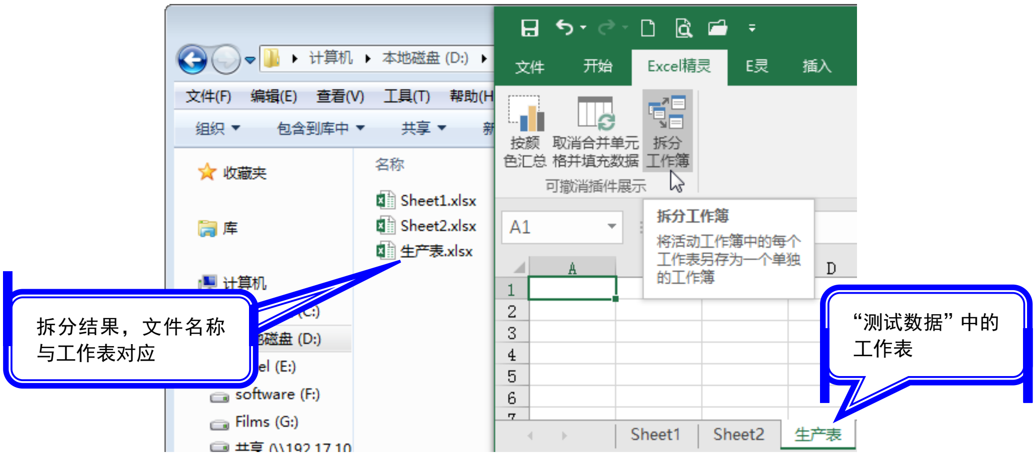


图 15–9　将工作表单独拆分成独立文件

15.3　课后思考

1．撤销功能可用于哪些领域?
2．撤销功能有哪些限制?
3．修改本书前面讲过的“创建工作表目录”过程，使其具备撤销功能。
4．修改本书第 14 章的插件“批量重命名.xlam”，使其具备撤销功能。
5．修改本书第 7 章的表格“将图片插到选区中.xlsm”，使其具备撤销功能。

第 16 章 使用 VSTO 设计插件的基本步骤

VSTO 的全称为 Visual Studio Tools For Office，即为在 Visual Studio 开发环境下的 Office 工具，可用于开发 Excel、Word、OutLook、PowerPiont、Project、Visio、InfoPath 等软件的插件，从而丰富 Office 功能。

本书以 Visual Studio Professional 2015 简体中文专业版作为演示工具，后续将其简称为 Visual Studio 2015。

在 Visual Studio 2015 中可以用 C#.net 和 VB.net 开发 Office 插件，由于 VB.net 的语法与 VBA 高度相似，在 VBA 代码的基础上小幅修改即可将代码应用于 VB.net 中，不需要学一门全新的语言，从而大大缩短学习时间，因此本书在讲解 VSTO 时以 VB.net 为蓝本，以 VBA 为基础，主要展示将 VBA 代码修改为 VSTO 代码的过程。在本书中与 VSTO 相关的 4 个章节不宜单独学习，而是必须先学习前面 15 章的 VBA 知识，然后通过后 4 章的讲解掌握修改方法，从而将 VBA 代码封装成 DLL 格式的 Excel 插件。

本章讲解 VSTO 的作用、安装 Visual Studio 2015 的步骤、使用 VSTO 开发插件的简易流程，以及将插件打包成安装程序的方法，后续的章节再进一步阐述将复杂 VBA 代码升级为 VSTO 代码的过程。

16.1 安装 Visual Studio 2015

Visual Studio 2015 文件较大，安装过程繁琐，不像 Office 那么简单，因此有必要专门讲解一下。

16.1.1 VSTO 对于 Excel 用户的意义

VSTO 是 Visual Studio 中的 Office 开发工具，可以用它开发 Office 插件，扩展 Office 功能。对于 Excel VBA 用户而言，它有什么意义呢?

尽管直接用 VBA 开发 Excel 插件比用 VSTO 方便得多，也快捷得多（由于支持录制宏从而既节约学习时间又节约开发时间，同时由于 VBA 代码是内置在工作簿中的，使用起来也方便得多），但是 VSTO 对于 Excel 用户而言仍然有着重要的价值。

第一，可以保护代码。VBA 的工程可以加密，但是仅需 1 秒钟就可以破解，同时 xlam 格式的插件不存在代码混淆工具或者加壳工具。而采用 VSTO 开发的 DLL 格式的插件被他人获得源代码的机会要小得多。

第二，如果打算设计商业插件，那么为了安装方便，同时让安装程序显示专业一些，也不宜使用 xlam 格式的加载宏，应该采用 VSTO 开发 DLL 格式的插件，然后再打包成 exe 格式的安装

程序。

第三，VSTO 开发的插件可以同时支持 Excel 和 WPS 表格，而 xlam 格式的加载宏无法在 WPS 中使用（WPS 个人版不支持 VBA）。

16.1.2　Visual Studio 版本介绍

VB 发展到 6.0 版本后就被微软放弃了，几年后推出了 Visual Studio 2002，内部集成了 VB.net 软件，版本号为 7.0，意思是 VB6.0 的升级版。

表 16-1 列出了已经上市的 Visual Studio 历代版本与年代。

表 16-1　Visual Studio名称、年代与编码对照表

软件名称	内部版本	软件名称	内部版本
Visual Studio 2002	7.0	Visual Studio 2012	11.0
Visual Studio 2003	7.1	Visual Studio 2013	12.0
Visual Studio 2005	8.0	Visual Studio 2015	14.0
Visual Studio 2008	9.0	Visual Studio 2017	15.0
Visual Studio 2010	10.0		

本书使用 Visual Studio 2015 作为演示工具，读者可以使用迅雷下载安装文件，也可以进入本书读者交流群 47700194，从群里下载软件以及注册码。下载网址如下所示。

http://download.microsoft.com/download/B/8/9/B898E46E-CBAE-4045-A8E2-2D33DD36F3C4/vs2015.pro_chs.iso

事实上，如果读者在使用 Visual Studio 2010，或者 Visual Studio 2017 也并不影响学习本书的内容，操作步骤是一致的，功能上变化不大，界面完全一致，高版本只是在录入代码的过程中更加智能化。

例如 Visual Studio 2015 相对于 Visual Studio 2010 新增了代码行号，并且在选择变量或者过程名称时会提示此变量/过程来自哪个模块，以及在模块中显示有多少引用等智能化的提示信息，而 Visual Studio Professional 2017 则又新增了一些智能的提示。

16.1.3　安装 Visual Studio 2015

软件是 ISO 格式的，下载后需要先解压，然后双击“vs_professional.exe”文件从而启动安装程序，具体的安装步骤如下：

1．双击“vs_professional.exe”文件后会弹出一个“正在初始化安装程序”的对话框，大约 1 分钟后进入“选择安装位置与安装类型”的界面，效果如图 16-1 所示。

2．安装位置采用默认值即可，但是安装类型需要选择“自定义”，然后单击右下方的“下一步”按钮。

3．进入“选择功能”界面后，需要勾选“Visual Studio 2015 更新”和“Microsoft Office 开发人员工具”复选框（C#.net 和 VB.net 属于必选项，不会出现在对话框中，因此不管如何选择都会安装 C#.net 和 VB.net 两个软件）,如图 16-2 所示。

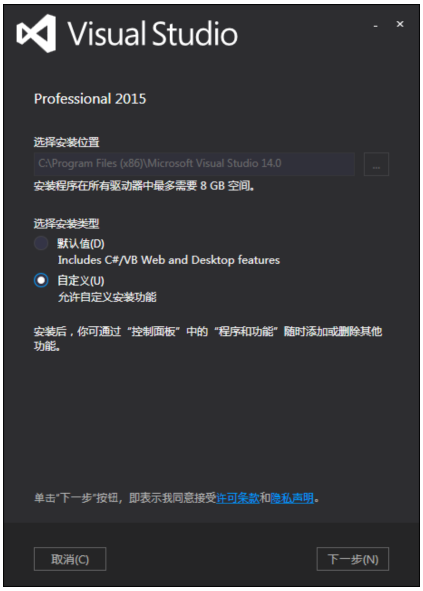


图 16–1　选择安装位置与安装类型

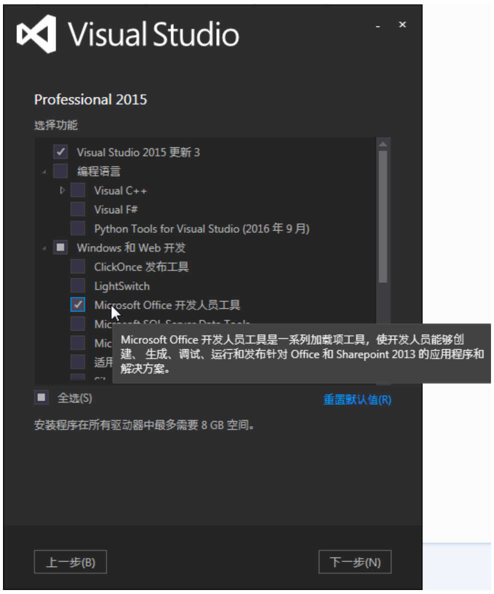


图 16–2　只选择必要的项目

4．单击"下一步"按钮后会弹出一个新的对话框，告知用户选择了哪些项目。此时继续单击"下一步"按钮，程序会开始安装，界面如图 16-3 所示。

安装过程可能会持续 2～3 个小时，根据电脑性能不同，安装时间长短也稍有不同。

5．安装完成后，在 windows 的开始菜单中可以看到"Visual Studio 2015"，单击即可启动 Visual Studio 2015 软件。

在首次启动时 Visual Studio 2015 会让用户选择开发环境，包含语言和配色。由于本书讲解 VB.net，因此选择"Visual Basic"，配色则按个人喜好选择即可，如图 16-4 所示。

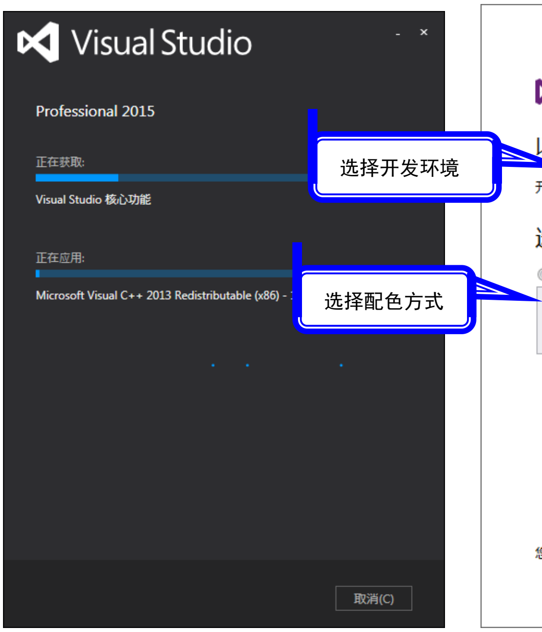


图 16–3　开始安装

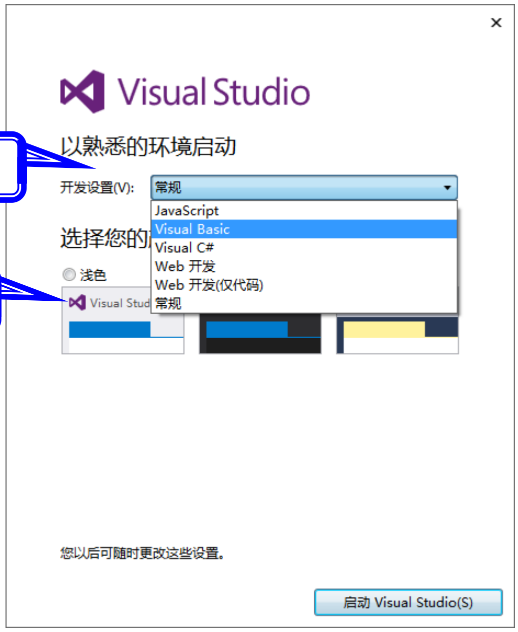


图 16–4　启动时选择开发语言

6．单击“启动 Visual Studio(S)”按钮，此时会进入 Visual Studio 2015 软件的起始页。

16.2　Excel 插件开发流程

本书并不是直接讲述利用 VB.net 开发 Excel 插件的，而是要求读者先学会 VBA 编程，然后再将 VBA 代码修改为符合 VB.net 语法的代码，从而简化 Visual Studio 2015 的学习难度。在这个过程中有两个重点，一是掌握 VBA 代码与 VSTO 代码的差异，二是掌握 VSTO 开发 Office 插件的基本步骤。为了让读者在阅读第 17 章内容时可以自行测试代码，特意将插件开发流程放在前面。

本节以一个简单的插件为例，展示插件的设计步骤。该插件包含两个菜单：一个名为“读取”，其功能是获取活动工作表的名称，一个名为“输出”，其功能是向活动单元格写入“VSTO”，插件名称为“玩转 VSTO”。

16.2.1　创建项目

本书介绍利用 VSTO 开发 Excel 插件，也就是外接程序，而非针对文档或者模板，具体步骤如下。

1．从 Windows 系统的“开始”菜单中单击“Visual Studio 2015”，从而打开 Visual Studio 2015 的起始页。也可以按【Win+R】组合键打开“运行”对话框，然后输入“devenv”，按下“确定”按钮即可。

2．选择“文件”命令→选择“新建项目”从而打开“新建项目”对话框。

3．选择“Office 外接程序”命令，然后在外接程序的类型列表中选择“Excel 2013 和 Excel 2016 VSTO 外接程序”命令，将上方的.Net Framework 版本号由 4.5.2 修改为 4（目的是让插件兼容 XP 系统和 Windows 7 系统），接着将名称修改为“玩转 VSTO”，将位置设置为 C 盘，如图 16-5 所示。最后单击“确定”按钮完成设置，此时会生成如图 16-6 所示的界面。

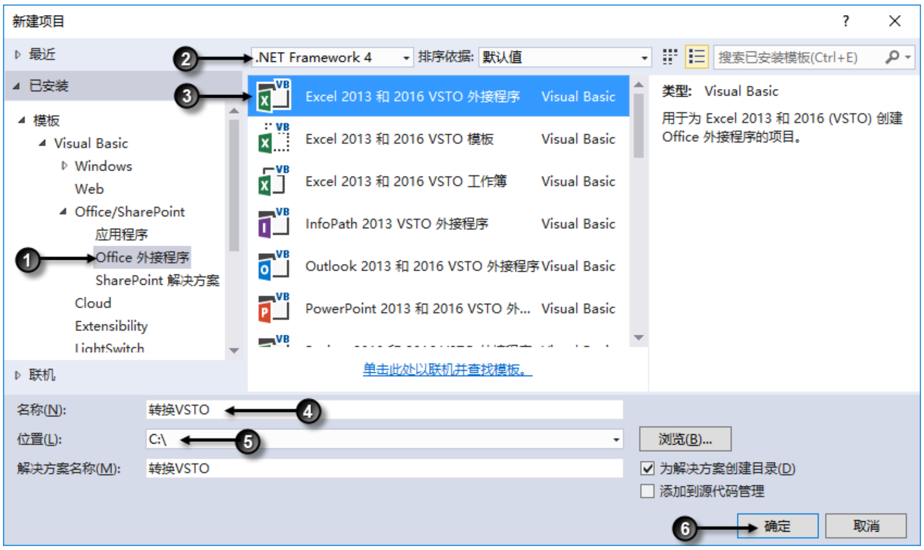


图 16-5　新建 Excel 的 VSTO 外接程序

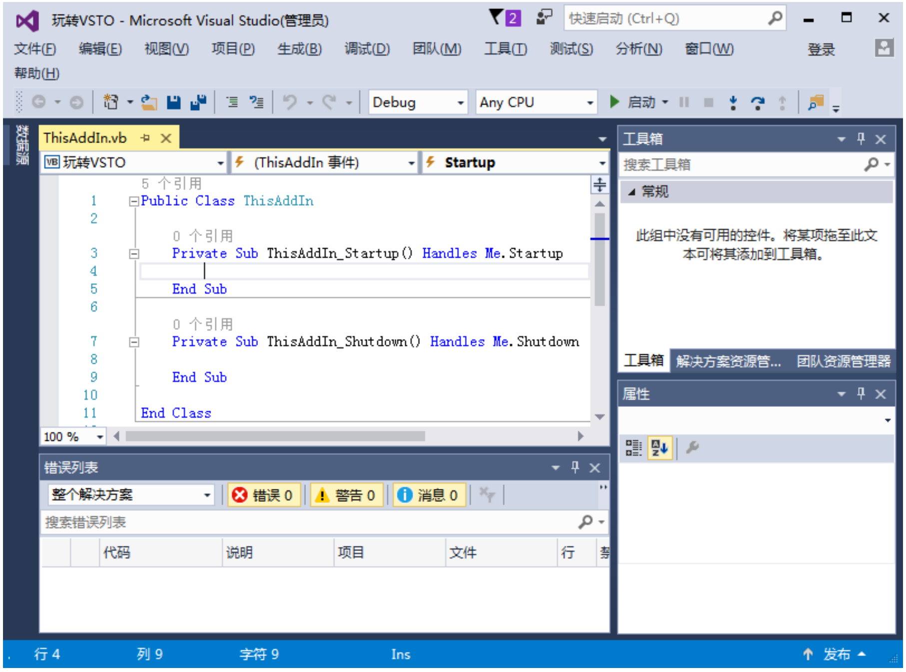


图 16–6　开发 Excel 外接程序的默认界面

图 16-6 中 ThisAddIn_Startup 事件在插件启动时触发，而 ThisAddIn_Shutdown 事件则在卸载插件时触发，ThisAddIn 则代表当前插件对象。

16.2.2　设计功能区

使用 VB.net 为 Excel 插件添加功能区按钮与 Custom UI Editor for Microsoft Office 2010 相比简单得多，不管是添加功能区组件还是回调，在 VB.Net 中都有极大的简化。

本例需要添加一个新的选项卡、一个组和两个命令按钮，具体步骤如下。

1．选择菜单“项目”命令→选择“添加组件”命令，然后选择最上方的“功能区（可视化设计器）”并单击添加按钮。

2．删除功能区组件自带的选项卡，如图 16-7 所示。

3．在右方的工具箱中找到 Tab 控件，并将它拖曳到左方的 Ribbon1 中，此时功能区中会产生一个名为 Tab1 的空白选项卡，效果如图 16-8 所示。

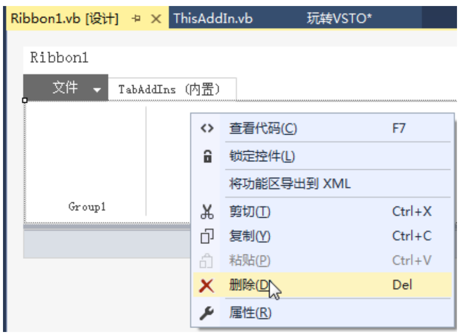


图 16–7　删除自带的选项卡

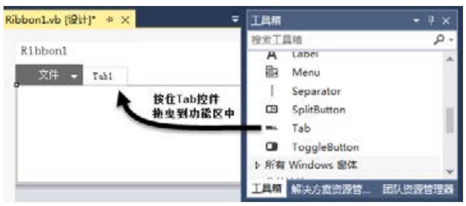


图 16–8　添加一个空白选项卡

4．在右下方的属性对话框中找到 Label 属性，并将默认的属性值“Tab1”修改为“玩转

VSTO”。

5．选择 Position 属性左方的田字图标，从而展开子项目，然后将 PositionType 属性的值修改为“BeforeOfficeId”，将 OfficeId 属性的值修改为“TabHome”，表示将“玩转 VSTO”选项卡的位置设置在“开始”选项卡之前。

6．返回工具箱，在“Office 功能区控件”列表中找到 Group 控件，并将它拖曳“玩转 VSTO”选项卡中，然后在属性对话框中将 Label 属性的值修改为“第一组”。

7．在工具箱中找到“Office 功能区控件”列表中的 Button 控件，并将它拖曳到“第一组”控件中，然后在属性对话框中将 Lable 属性的值修改为“读取”，将 OfficeImageId 属性的值修改为“A”，再将 ControlSize 属性的值由“RibbonControlSizeRegular”修改为“RibbonControlSizeLarge”，表示将按钮“读取”显示为大图标，图标为字母 A（在实际运行时将显示图标效果）。

8．重复步骤 7，添加第 2 个按钮，并且将它的 Label 属性的值设置为“输出”，将 OfficeImageId 属性的值设置为“B”，此时功能区的布局如图 16-9 所示。

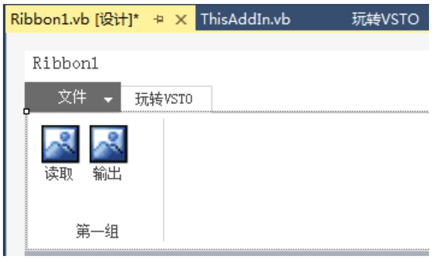


图 16–9　功能区布局预览

16.2.3　写入 Sub 过程

本例中有两个按钮，因此需要编写两个 Sub 过程。同时，为了让 VSTO 与 Excel 的对象能互相通讯，还需要创建一个代表 Excel 应用程序的公共变量，然后通过这个变量引用 Excel 的所有子对象，具体步骤如下。

1．选择菜单“项目”→“添加模块”，然后单击“添加”按钮，此时软件会自动打开 Module1。

2．模块中默认有以下两句代码：

```
Module Module1

End Module
```

需要在两句代码中间插入以下代码，表示声明一个名为 app 的公式变量，并且在声明的同时对它赋值为 Globals.ThisAddIn.Application。

```
Public app As Excel.Application = Globals.ThisAddIn.Application
```

代码中 ThisAddIn 代表当前插件本身，ThisAddIn.Application 则代表 Excel。

3．在上方的导航栏中双击“Ribbon1.vb[设计]”，然后双击“读取”按钮，从而进入功能区的代码窗口。在代码窗口中默认产生的代码如图 16-10 所示。

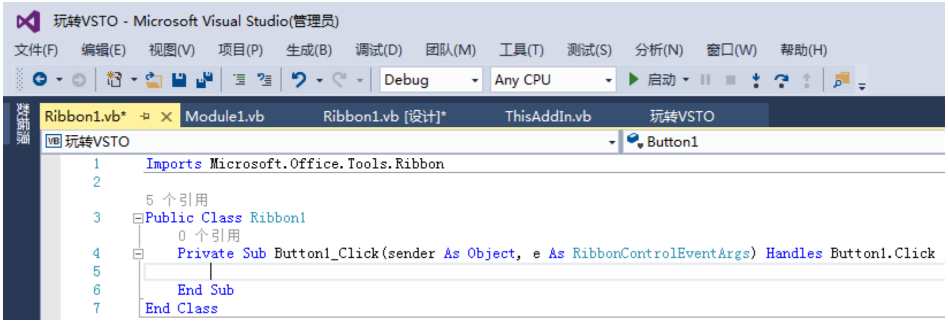


图 16-10　功能区的代码窗口

在“End Sub”上方插入以下代码，表示在单击 Button1 时报告 Excel 的活动工作表名称。

```
MsgBox(app.ActiveSheet.name)
```

在 VSTO 中所有参数都必须使用括号，但在 VBA 中不需要括号。另一个差异是 VSTO 在引用 Excel 的一切对象时都需要在前面加“app.”。

4．返回“Ribbon1.vb[设计]”窗口，然后双击“输出”按钮，此时程序会进入功能区的代码窗口，并且生成 Button2 按钮的 Click 事件过程外壳，代码如下：

```
Private Sub Button2_Click(sender As Object, e As RibbonControlEventArgs) Handles Button2.Click

End Sub
```

在“End Sub”上方插入以下代码，表示单击 Button2 时向 Excel 的 A1 单元格写入值：

```
app.Range("a1").Value = "转换 VSTO"
```

16.2.4　生成 DLL 插件

通过前面的步骤，已经完成了插件的功能设计，接下来的工作就是生成插件。

选择菜单“生成”命令→“选择“生成玩转 VSTO”命令，工作结束。

此时如果在下方的“错误列表”中产生了“无法在当前用户的 Windows 证书存储中找到代码签名证书。若要更正此问题，需要禁用 ClickOnce 清单的签名或将证书安装到证书存储中。”，那么可以按以下步骤操作。

1．选择菜单“项目”命令→选择“玩转 VSTO 属性”。

2．单击“创建测试证书”命令，然后在弹出的对话框中录入密码两次，密码随意，此处以 123 为例，界面如图 16-11 所示。

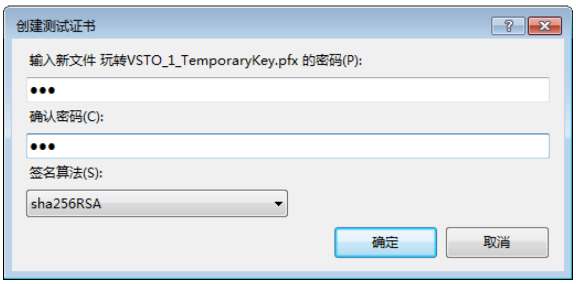


图 16-11　创建测试证书

3．再次选择菜单“项目”命令→选择“玩转 VSTO 属性”命令，然后关闭 Visual Studio 2015。

4．打开 Excel 2016，此时可以在功能区中看到生成的选项卡“玩转 VSTO”，单击菜单“读取”，将会弹出活动工作表名称，效果如图 16-12 所示，表明插件设计成功。

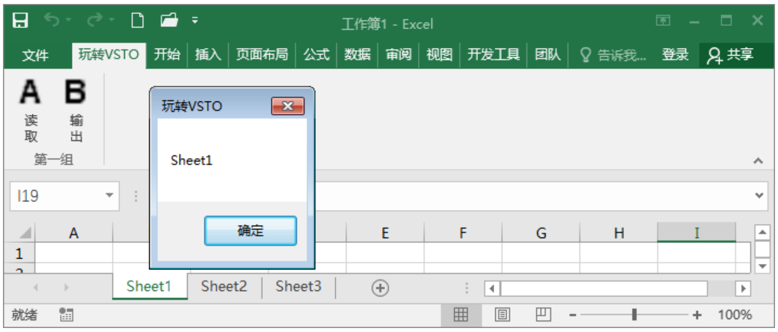


图 16–12　执行命令“读取”

如果选择菜单“输出”，那么 A1 单元格中会产生“玩转 VSTO”字样。

16.3　将插件打包成安装程序

单独的 DLL 格式的插件无法直接发给客户使用，必须将它打包成 EXE 格式的安装程序。本节介绍使用 Inno Setup 软件打包插件的方法。

16.3.1　Inno Setup 软件介绍

Inno Setup 一款免费的、开源的、专用于制作安装包的工具，目前中文版的最高版本号是 5.5.9，可以从以下网址下载（随时有可能失效，若已失效请自行上网下载）:

http://www.jz5u.com/Soft/softdown.asp?softid=21812

下载后需要用 Winrar 或者其他解压工具将它解压，然后双击安装软件，一直单击“下一步”按钮直到安装结束。完成后在 Windows 的开始菜单能看到“Inno Setup 5”或者“Inno Setup 编译器”，单击“Inno Setup 编译器”文件即可打开此软件的欢迎界面。

16.3.2　打包插件安装程序

Excel VBA 设计的加载宏一般只有一个文件，该文件要么是 xla 格式，要么是 xlam 格式。VSTO 设计的插件则包含 10 多个文件，如果有外部引用，那么还会生成更多文件。

在本例使用 VB.net 设计好 Excel 插件后，文件会生成到以下路径：

C:\玩转 VSTO\玩转 VSTO\bin\Debug

此路径下有 16 个文件，其中与“玩转 VSTO”相关的文件 5 个，其他都是辅助文件。需要打包到 EXE 安装程序中的文件共用有 9 个，包含：

Microsoft.Office.Tools.Common.dll

Microsoft.Office.Tools.Common.v4.0.Utilities.dll

Microsoft.Office.Tools.dll

Microsoft.Office.Tools.Excel.dll

Microsoft.Office.Tools.v4.0.Framework.dll

Microsoft.VisualStudio.Tools.Applications.Runtime.dll

玩转 VSTO.dll

玩转 VSTO.dll.manifest

玩转 VSTO.vsto

其中前 6 个文件是固定的，在设计插件时总会生成这 6 个固定名称的文件，后 3 个文件则由创建项目时取的名称而定。

要将 VSTO 生成的文件打包成 EXE 的安装程序，需要准备以下 3 个额外的文件。

一个 ICO 格式的图标文件，可以从网上随意下载，也可以自己用 PhotoShop 设计。在生成安装文件时，将用它来定义安装程序的图标。此文件是非必要的文件，如果不提供，那么 Inno Setup 会调用默认的图标。

一个名为 WizModernImage.bmp 的 164*314 的 bmp 格式的图片，用于美化安装程序的安装界面。这也是一个非必要的文件，如果不提供，Inno Setup 会调用默认的图片文件。

一个名为安装文件模板.iss 的模板，由于模板的设计过程较为复杂，而且其代码采用的是 delphi 语言，与 VB.net 大大不同，因此本书不再一一介绍代码的设计过程，而是直接提供一个模板文件给读者，读者只需要替换一下文件名称即可使用。该文件可通过以下路径找到：

本例案例文件请参考：..\ 第 16 章\安装文件模板.iss

或者

本例案例文件请参考：..\ 第 16 章\玩转 VSTO\玩转 VSTO\bin\Debug\安装文件模板.iss

在“安装文件模板.iss”中有以下代码。

两行代码定义了两个常量，第一行代表安装程序的名称，第二行代表含版本号的程序名称。

```
#define MyAppName "玩转 VSTO"
#define MyAppVerName "玩转 VSTO 1.0"
```

以下代码分别定义了安装程序的名称、默认的安装路径、显示在开始菜单中的文件夹名称、默认的安装文件存放路径、安装文件图标、安装文件名称，以及安装界面的图片。

```
[Setup]
AppName={#MyAppName}
AppVerName={#MyAppVerName}
VersionInfoProductName=玩转 VSTO
DefaultDirName= "C:\Program Files\玩转 VSTO"
DisableDirPage=no
DisableProgramGroupPage=yes
DefaultGroupName=玩转 VSTO
OutputDir=.
SetupIconFile=1.ico
OutputBaseFilename=setup
WindowShowCaption=no
DisableWelcomePage=no
WizardImageFile=WizModernImage.bmp
```

其中“OutputDir=.”表示安装文件产生在“安装文件模板.iss”的相同路径下；“SetupIconFile=1.ico”表示安装文件的图标来自名为 1.ico 的图标文件。

以下代码表示在安装程序中的第一个界面中显示几句提示信息，属于非必要的信息。

```
[Messages]
SetupWindowTitle=玩转 VSTO 安装向导
ClickNext=单击“下一步”按钮继续，或单击“取消”按钮退出安装程序。%n%n%n%n%n 为确保本软件能一次安装成功，请尽可能先关闭 360 或者电脑管家、金山毒霸之类的安全软件，然后安装本软件。
```

以下代码表示让安装程序显示为中文界面，若改为“Name: "en"; MessagesFile:

"compiler:English.isl"”则会显示为英文界面。

```
[Languages]
Name: "chinesesimp"; MessagesFile: "compiler:Default.isl"
```

以下代码表示在 Windows 的开始菜单中显示一个卸载当前插件的快捷方式。

```
[Icons]
Name: "{group}\{cm:UninstallProgram,{#MyAppName}}"; Filename: "{uninstallexe}"
```

以下代码表示安装软件时将以上 9 个文件复制到用户选择的路径下。“{app}”代表用户在安装过程中选择的路径。

```
[Files]
Source:"Microsoft.Office.Tools.Common.dll"; DestDir: "{app}"; Flags: ignoreversion
Source:"Microsoft.Office.Tools.Common.v4.0.Utilities.dll"; DestDir: "{app}"; Flags: ignoreversion
Source:"Microsoft.Office.Tools.dll"; DestDir: "{app}"; Flags: ignoreversion
Source:"Microsoft.Office.Tools.Excel.dll"; DestDir: "{app}"; Flags: ignoreversion
Source:"Microsoft.Office.Tools.v4.0.Framework.dll"; DestDir: "{app}"; Flags: ignoreversion
Source:"Microsoft.VisualStudio.Tools.Applications.Runtime.dll"; DestDir: "{app}"; Flags: ignoreversion
Source:"玩转 VSTO.dll"; DestDir: "{app}"; Flags: ignoreversion
Source:"玩转 VSTO.dll.manifest"; DestDir: "{app}"; Flags: ignoreversion
Source:"玩转 VSTO.vsto"; DestDir: "{app}"; Flags: ignoreversion
```

以下代码表示向注册表中写入插件的相关信息，其中 Subkey 代表路径，“{code:pathSS}”表示调用函数 pathSS，通过该函数执行写入。

```
[Registry]
Root: HKCU; Subkey: "Software\Microsoft\玩转 VSTO"; ValueType: string; ValueName: "SetupPath";
ValueData: {code:pathSS}； Flags: uninsdeletevalue
```

以下代码表示在安装界面出现之前弹出一个提示框，告知用户一些必要的信息。如果只在 Windows 10 系统中使用本插件，那么可以取消这些提示。

```
[Code]
function InitializeSetup(): Boolean;
begin
  Log('InitializeSetup called');
  Result := MsgBox('友情提示:'#13'1.如果您的操作系统是Win 10或者Win 8，而且Excel是2013版或者2016
版，可以直接安装“Setup.exe”。'#13'2.操作系统或者 Excel 不属于以上版本，请先安装“运行环境.exe”，后
安装“Setup.exe”。'#13''#13'需要先装“运行环境.exe”请点“否”，然后请双击”运行环境.exe“;如果已经安
装运行环境或者不需要安装运行环境请点“是”', mbConfirmation, MB_YESNO) = idYes;
  if Result = False then
end;
```

以下代码定义了一个 pathSS 的函数，通过函数向注册表中写入插件信息。

```
   function   pathSS(Param: String): string;
begin
    result:=  ExpandConstant('{app}')
     RegWriteStringValue(HKEY_CURRENT_USER,'Software\KingSoft\Office\ET\AddinsWL','玩转 VSTO','')
     RegWriteStringValue(HKEY_CURRENT_USER, 'Software\Microsoft\Office\Excel\Addins\玩转 VSTO',
'Description', '玩转 VSTO');
     RegWriteStringValue(HKEY_CURRENT_USER, 'Software\Microsoft\Office\Excel\Addins\玩转 VSTO',
'FriendlyName', '玩转 VSTO');
     RegWriteDWordValue(HKEY_CURRENT_USER, 'Software\Microsoft\Office\Excel\Addins\玩转 VSTO',
'LoadBehavior', 3);
     RegWriteStringValue(HKEY_CURRENT_USER, 'Software\Microsoft\Office\Excel\Addins\玩转 VSTO',
'Manifest',ExpandConstant('{app}')+'\玩转 VSTO.vsto|vstolocal');
 end;
```

其中第一句 RegWriteStringValue 表示将插件注册到 WPS 的 ET 表格软件中去，从而使本插

件可用于 WPS，后 4 句代码则表示将插件注册到 Excel 中。

以下代码表示在卸载插件时清除注册表中的注册信息。

```
procedure CurUninstallStepChanged (CurUninstallStep: TUninstallStep );
begin
    RegDeleteKeyIncludingSubkeys(HKEY_CURRENT_USER, 'Software\Microsoft\Office\Excel\Addins\ 玩 转
VSTO');
end;
```

以上所有代码中的“玩转 VSTO”都来自插件的名称。

以上 3 个文件(ico 图标文件、WizModernImage.bmp 安装界面背景图片和安装文件模板.iss)在存放到“C:\玩转 VSTO\玩转 VSTO\bin\Debug”路径下之后，双击打开安装文件模板.iss，然后选择菜单“构建”→“编译”，大约两三秒钟后 Inno Setup 会在 Debug 文件夹中生成一个名为 Setup.exe 的文件，其图标等同于 1.ICO 文件，效果如图 16-13 所示。

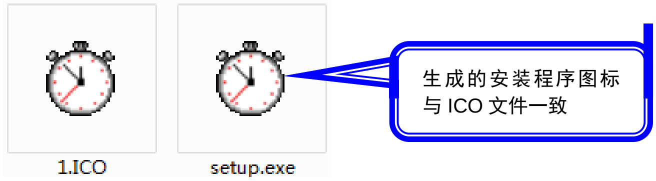


图 16–13　Inno Setup 生成的安装程序

如果读者要将另一个插件打包，那么直接将“安装文件模板.iss”中的插件名称“玩转 VSTO”批量替换新的插件名称即可，两三秒钟完成，具体步骤如下。

利用 VB.net 生成插件后，替换“安装文件模板.iss”中的插件名称，然后将此文件复制到该插件的 Debug 目录中，如果有必要，也一并将“1.ICO”和“WizModernImage.bmp”文件一并复制进去，当然针对不同的插件应该使用不同的图标和背景图片，在复制之前可以稍加修改。

复制 3 个文件进去后，双击“安装文件模板.iss”，完成插件打包工作。

16.3.3　安装插件

16.3.2 节中生成的 Setup.exe 属于插件的安装程序，只要双击它即可启动安装程序，然后单击几次“下一步”按钮便可以完成安装工作。不过在安装之前有必要补充一个关于软件兼容性的知识。

使用.net 框架的软件或者插件时，在电脑中必须有与 4.0 版或者更高版本的.net framework 软件，否则程序无法运行。Windows 7 自带的.net 框架是 3.5 版、Windows XP 自带的.net 框架是 1.0 版，它们版本太低，不足以满足需求，因此如果要求 Visual Studio 2015 开发的插件兼容 Windows 7 或者 Windows XP，那么需要让用户先安装.net framework 4.0 或者 4.5 版，而 Windows 8 和 Windows 10 的用户则不需要这么操作，因为 Windows 8 内部集成了 4.0 版的.net 框架，Windows 10 集成了更高版本的.net 框架。

此外，若要使用 Visual Studio 2015 开发的 Office 插件，当 Office 版本低于 2013 时，还要在 Windows 中安装一个 Microsoft visual studio 2010 tools for office runtime 工具，否则插件无法运行。而 Office 2013 和 2016 都集成了这个工具，因此不需要安装此工具。同时，此工具还分 32 位版本和 64 位版本，软件会自动判断版本号，从而选择安装相应版本的工具。

简而言之，Windows 8 及以上系统、Office 2013 及以上版本的用户直接安装 Office 插件本身即可，而低版本操作系统或者低版本 Office 的用户则需要安装额外的辅助软件。

笔者已经将这 3 个辅助软件打包成一个“运行环境.exe”，如果读者需要让自己的插件兼容老版本的 Windows 和 Office，那么应该将此文件与前面生成的“Setup.exe”文件放在一起（通常的做法是使用压缩软件压缩成一个 rar 格式的文件），用户在安装插件时会根据自己的实际情况决定直接安装“Setup.exe”，还要先安装“运行环境.exe”。

假设本机的配置是 Windows10 和 Office 2016，便不需要安装“运行环境.exe”，直接安装“setup.exe”即可，具体步骤如下。

1．双击“Setup.exe”，安装程序会弹出如图 16-14 所示的提示信息。

2．单击 “是”按钮，进入如图 16-15 所示的欢迎界面。

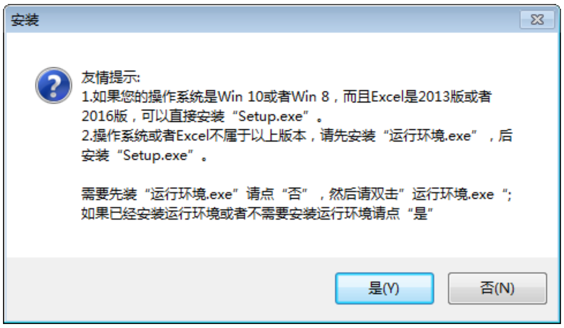


图 16–14　安装前的提示



图 16–15　安装程序的欢迎界面

3．单击“下一步”按钮进入“选择目标位置”界面，在此处可以将默认的路径修改为其他路径（见图 16-6），但是最佳操作方式是保持默认值不变，直接单击“下一步”按钮进入“准备安装”界面。

4．单击“安装”按钮，程序会一两秒钟后安装完成，从而进入“安装向导完成”界面，如图 16-17 所示。

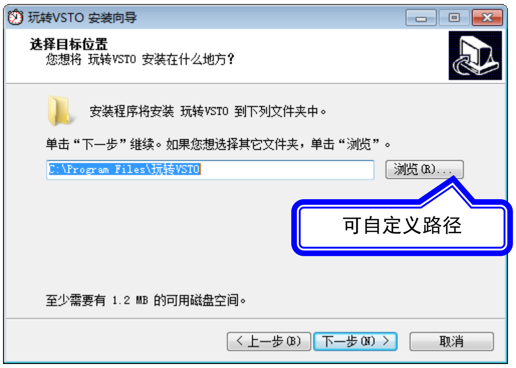


图 16–16　“选择目标位置”界面



图 16–17　“安装向导完成”界面

安装完成后，启动 Excel 时会弹出以下对话框，单击“安装”按钮，如图 16-18 所示。

安装后，在 Excel 中会产生如图 16-19 所示的功能区选项卡，表示安装成功。如果将插件安装在 WPS 上，则会看到如图 16-20 所示的界面。

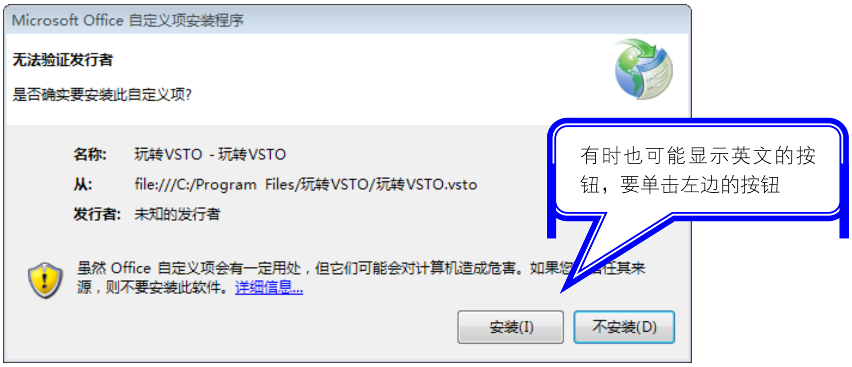


图 16–18　首次启动 Excel 时的安装选项

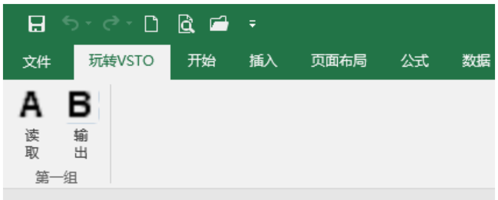


图 16–19　安装插件后的 Excel 界面



图 16–20　安装插件后的 WPS 界面

由于开发插件时采用的是 Excel 内部图标，而 WPS 中只有极少数图标和 Excel 的图标重名，因此若在设计菜单按钮时采用了 Excel 的内部图标名称，就有可能导致插件在 WPS 中显示为问号。假设需要让插件同时兼容 Excel 和 WPS，则应该采用自定义的图标。

本例案例文件请参考：..\ 第 16 章\玩转 VSTO\玩转 VSTO\bin\Debug

16.4　课后思考

1．什么是 VSTO?
2．为什么要封装插件?
3．使用 VB.net 制作功能区比用“Custom UI Editor for Microsoft Office 2010”有何优势?
4．VSTO 与 VBA 中对变量赋值时有何差异?
5．在 VSTO 中引用 Excel 对象有何注意事项?

第 17 章　VSTO 与 VBA 的差异

本书的思路是先学习 VBA，然后将 VBA 代码修改为 VSTO 代码，从而减少约 8 成的工作量，因此在修改代码前有必要了解 VSTO 和 VBA 代码在语法上的差异，以及调试代码的差异。

本章分 9 个方面逐一阐述。

17.1　变量、常量与数据类型

变量、常量与数据类型是编程的基础，本节逐一分析 VSTO 中的变量、常量、数据类型相对于 VBA 有哪些变化，应如何修改。

对于 VSTO 和 VBA 的差异，本章采用“VS+×××”方式编号，在后续修改代码时可以直接采用编号来说明修改依据，从而简化描述过程。

17.1.1　数据类型

VSTO 相对于 VBA 的数据类型大同小异，例如两者都有 Byte 型，取值范围都是 0~255，再如都有 Double 型，且取值范围都相同。本节主要讲述二者不同又易混淆之处。

VS001：VSTO 中取消了 Currency 类型，当 VBA 代码中有用到 Currency 类型时，将它修改为 Double 就好。

VS002：VSTO 中的 Short 等同于 VBA 中的 Integer；VSTO 中的 Integer 等同于 VBA 中的 Long；VSTO 中的 Long 远远大于 VBA 中的 Long，其取值范围在–9223372036854775808 到 9223372036854775807 之间。

在 VSTO 中，Short 可简写为 int16，Integer 可简写为 int32，Long 可简写为 int64，这样比较便于记忆和识别。

VS003：VSTO 中取消了变体型 Variant，一切值和对象都是 Object，就连字符串、数值都是对象，有自己的方法。修改 VBA 代码在遇到 Variant 时必须将其修改为 Object。

VS004：当声明 Excel 的对象变量时，必须加前置的“Excel.”。例如:

```
Dim Rng as Excel.Range
```

但是在代码中直接调用 Excel 的对象时不是加前置的“Excel.”，而是前置“app.”。其中 app 是变量，由自己的命名而定。例如在模块中用以下方式声明一个公共变量，它代表 Excel 对象，后续引用 Excel 对象时都需要借用此变量来引用对象。

```
Public app As Excel.Application = Globals.ThisAddIn.Application
```

例如调用 Range ("A1")时要改用 app. Range ("A1")。

不过对于以下代码却不能在 Application 前添加“App.”，而是要将 Application 修改为 app，因为 Application 即为 Excel 对象，它不再有父对象。

```
Application.ScreenUpdating = False
```

VS005：在 VSTO 中对对象的属性赋值时必须指定属性的名称，否则无法赋值。Visual Studio

2015 会有错误提示，而 Visual Studio 2010 则没有错误提示，如果忘记写属性名称则会导致程序赋值不成功，将给代码查错带来很大的障碍。例如：对单元格 A1 赋值为 123，VBA 中可以用代码“Range("A1")=123”，忽略属性名称，而在 VSTO 中则必须采用代码“app.Range("A1").Value = 123”，不再支持默认属性 Value。读者在测试代码时一定要注意这一点，当把 VBA 代码复制到 VSTO 后将 Range 对象的属性值补充完整。

同理，窗体中的文本框控件也不再有默认属性，在对文本框赋值时需要写明 Text 属性名称；选项按钮 RadioButton 不再有默认属性，调用时需注明 Checked 属性名称。

17.1.2　变量

VSTO 对变量的管理和 VBA 稍有不同，具体表现在以下几点。

VS006：在 VSTO 中声明变量时，如果前一个变量未指定数据类型，后一个变量指定了数据类型，那么前一个变量将会声明为后一个变量的相同类型。因此 VBA 中的“Dim Arr, Mystr As String”到了 VSTO 中应该写为“Dim Mystr As String, Arr”，也可以写为“Dim Arr As Object, Mystr As String”。

VS007：变量必须先声明才能调用，否则代码无法执行。不过 VSTO 同时也提供了便利：在声明变量的同时可以赋值，不像 VBA 那样声明和赋值必须用两句代码。

VS008：变量必须先赋值，然后才能调用。例如以下代码在 VBA 中可以正常运行，VSTO 中则不行，需要将“i As long”修改为“i As long = 0”。

```
Sub test()
    Dim cell As Excel.Range, i As long
    For Each cell In app.Selection
        If cell.Value > 10 Then i = i + 1
    Next cell
    MsgBox ( "大于 10 的单元格有" & i & "个" )
End sub
```

VS009：在对对象变量赋值时不需要使用 Set。

VS010：在变量的数据类型不同时，两个变量（也可以是一个变量和一个对象）不能做比较。例如以下代码，如果 A1 的值是文本，那么“Rng.Value > Item”无法获得比较结果。

```
Dim Item As Integer, Rng As Excel.Range
Item = 2
Rng = app.Range("A1")
MsgBox(Rng.Value > Item)
```

再如以下代码在 VBA 中可以正常比较，而在 VSTO 中则无法比较。

```
Dim a
a = 123
MsgBox a = "ABC"
```

解决办法有两个，如果要比较的对象不是文本，那么可以使用 ToString 方法将数据变量转换成文本后再做比较，如果要比较的对象是数值，那么使用 If 函数判断对象变量是不是数值，如果不是数值就忽略掉，从而完美解决问题。

因此上面第二段代码的修改方式为“MsgBox a .ToString= "ABC"”。

VS011：在闭合的空间内声明的变量不能在此空间以外调用。例如在以下过程中 If 和 End if 形成了一个闭合的空间，在此空间声明的变量不能在 End If 以后调用。

```
Dim aa As Integer
aa = app.Range("A1").Value
If aa >= 0 Then
```

```
        Dim bb As Integer
        bb = aa * 10
    Else
        Dim cc As Integer
        cc = aa / 10
    End If
    MsgBox(bb + cc)
```

此外，With…End with 语句和 Select Case… End Select 语句都会形成闭合空间。

17.1.3　常量

这里讲的常量不是针对 Const 语句声明的常量，而是指调用 VSTO 的内部常量。

VS012：在 VBA 中调用 VBA 的内部常量和 Excel 的内部常量都直接写常量名称即可，而在 VSTO 中调用 Excel 的一切内置常量都不能直接写常量名称，要么写该常量对应的数值，要么对常量加上父对象，表明当前常量来自哪一个类，从而便于识别和理解，但是书写上就复杂多了。不管使用数值还是添加父对象都需要使用相同办法去搜索目标，因此本节主要介绍搜索目标的步骤。

假设要把以下代码改成 VSTO 代码：

```
Dim Rng As Range
Set Rng = Cells.Find(What:="中国", After:=ActiveCell, LookIn:=xlFormulas, LookAt:=xlPart)
```

最先想到的修改结果可能是这样：

```
Dim Rng As Excel.Range = app.Cells.Find(What:="中国", After:=app.ActiveCell,
LookIn:=xlFormulas, LookAt:=xlPart)
```

但是很显然，代码无法运行，因为代码中用到了两个 Execl 的常量——xlFormulas 和 xlPart，并未指定该常量的父对象。

以 xlFormulas 为例，搜索其父对象的方法是：按下 F2 键打开对象浏览器，然后在左上方的搜索框中录入 xlFormulas 并按下 Enter 键，右方会弹出 xlFormulas 相关的信息，包含它的父对象名称和数值，效果如图 17-1 所示。

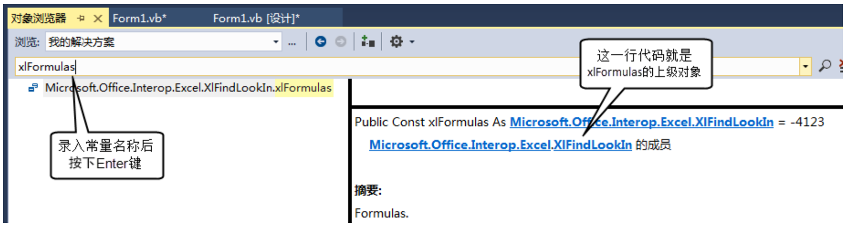


图 17–1　搜索常量的上级对象

因此，xlFormulas 的完整写法是“Microsoft.Office.Interop.Excel. XlFindLookIn.xlFormulas”。

同时，图 17-1 中显示了 xlFormulas 的常数值为-4123，因此也可以直接写-4123。

用同样的方法可以找到常量 xlPart 的父对象名称，其完整书写方式为“Microsoft.Office. Interop.Excel.XlLookAt.xlPart”。

17.2　函数

在 VSTO 中可以调用 Excel 函数，但是和 VBA 调用工作表函数稍有区别，而且还删除了一些函数。如果不了解这些差异，那么将 VBA 代码改成 VSTO 代码后无法执行。

17.2.1　调用方式不同

VS013：在 VBA 中调用工作表函数需要加前置的“Worksheetfunction.”，而调用 VBA 的内部函数则直接写函数名称或者加前置的“VBA.”。在 VSTO 中调用 Excel 的工作表函数和在 VBA 中的操作方式完全相同，但调用 VBA 函数则不能通过“VBA.”实现。

在 VSTO 中将内部函数分为数值运算函数和文本运算函数，调用数值运算函数可采用前置的“Math.”，而调用文本运算函数则采用前置的“Strings.”。图 17-2 和图 17-3 展示了调用过程。

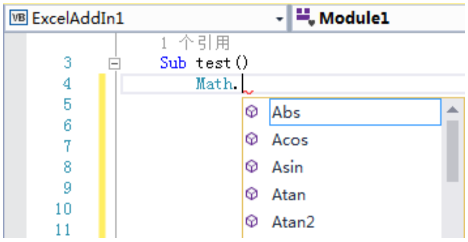


图 17–2　调用数值运算函数

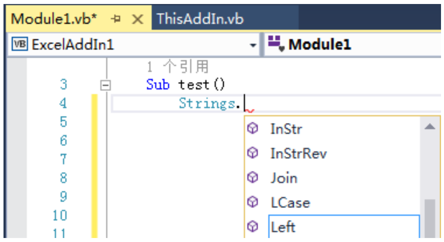


图 17–3　调用文本运算函数

在模块中调用 Left 或者 Right 函数可以直接书写函数名称，但是在窗体中调用 Left 或者 Right 函数则必须使用 Strings.Left 或者 Strings.Right。

VS014：VBA 中的参数有时需要加括号，有时不需要加括号；在 VSTO 中，任何时候都需要加括号，否则代码无法运行，如下所示。

VBA：

```
MsgBox Sheets.Count
Sheets.Add Sheets(Sheets.Count)
```

VSTO：

```
MsgBox(app.Sheets.Count)
app.Sheets.Add(app.Sheets(app.Sheets.Count))
```

17.2.2　函数差异

VS015：VSTO 使用 StrDup 函数取代了 VBA 中的 string 函数，例如：

StrDup(5, "P")的代码运行结果是“PPPPP”。

VS016：在 VBA 中可以通过“VBA.DoEvents”方式调用 DoEvents 函数，而在 VSTO 中要改用“System.Windows.Forms.Application.DoEvents”。

VS017：在 VSTO 中取消了 Lenb、Leftb、Rightb 等函数，其中 Lenb 可用 GetByteCount 取代，其他两个函数则不存在功能相同或相近的函数。

```
Dim mystr As String = "中国人 Ab"
MsgBox(Len(mystr))     '结果为 5
MsgBox(System.Text.Encoding.Default.GetByteCount(mystr)) '结果为 8
```

VS018：Shell 在 VSTO 中的语法有较大变化，语法如下：

```
Public Function Shell( ByVal PathName As String, Optional ByVal Style As AppWinStyle = AppWinStyle.MinimizedFocus, Optional ByVal Wait As Boolean = False, Optional ByVal Timeout As Integer = -1 ) As Integer
```

其中第一参数代表应用程序，第二参数代表处理窗口的方式，第三参数代表是否等待代码执行完成，赋值为 True 表示等待，第四参数表示等待时间。具体用法如下：

```
ID = Shell("""C:\Program Files\display.exe"" -a -q", , True, 100000)
```

这里的 100000 是指如果程序大于 100000 毫秒尚未执行完毕，那么只等待 100000 毫秒，就

收回控制权。如果不写时间，那么表示一直等待下去，直到调用的程序执行完。

VS019：VSTO 提供了专用的打开网页的方法 Process.Start，不再需要 Shell 函数。当然，VBA 原本的 Shell 函数仍然继续支持。以下代码用于打开网址 http://excelbbx.net：

```
System.Diagnostics.Process.Start("http://excelbbx.net")
```

Process.Start 方法支持两个参数，第一参数是程序名称，第二参数是程序的参数。例如要打开 C 盘，然后选中名为 05.bmp 的文件，那么可使用以下代码：

```
System.Diagnostics.Process.Start("explorer.exe", "/select, c:\05.bmp")
```

VS020：日期的计算方式有较大的变化，在 VBA 中使用加号或者减号即可计算，而 VSTO 则需要使用比较复杂的算法。

日期相减采用 Subtract 函数。以上代码表示用现在的日期减去 2012 年 8 月 13 日的天数差异，即对两个日期值求差。

```
Dim D As Date = #8/13/2012#
I = Now.Subtract(D).TotalDays
```

以下代码表示对指定的日期 2015 年 8 月 13 日累加 2 天，允许累加值为负数。

```
Dim aa As Date = #8/9/2015#
MsgBox(aa.AddDays(2))
```

以下代码表示在 2012 年 5 月 9 日基础上累加 2 个月，从而生成一个新的日期。

```
MsgBox(DateAdd("M", 2, "2012 年 5 月 9 日"))
```

VS021：VBA 有 Date 函数，用于生成今天的日期，也有 Date 语句，用于修改电脑右下角显示的日期。在 VSTO 中取消 Date 函数和 Date 语句，但提供了一个 Date 关键字，可以通过它调用与日期相关的一些函数和方法，例如 Date.Today、Date.Now、Date. IsLeapYear、Date.DaysInMonth 和 Date.Parse 等。

VS022：使用 Format 函数格式化日期时，在 VBA 中使用 m 代表月，而在 VSTO 中小写的 m 代表分钟，大写的 M 才代表月份，因此在 VSTO 中必须按以下方式录入代码：

```
Dim aa As Date = #8/9/2015#
MsgBox(Format(aa, "yyyy-M-d"))
MsgBox(Format(aa, "yyyy 年 MM 月 dd 日"))
```

17.3 数组

VSTO 对数组的处理能力比 VBA 强大得多，二者差异也较大，必须掌握这些差异才能让 VBA 代码在 Visual Studio 平台中顺利运行。

17.3.1 原本功能的差异

VS023：在 VBA 中数组的下标可以是任意值，在定义变量时可以随意指定。VSTO 中的数组变量不允许下标大于 0，例如在“Dim Arr(1 To 5), Arr2(3 To 5)”中的两个数组变量都不可能声明成功。不过如果引用区域的值从而生成数组，那么该数据的下标则是 1。

还有使用 GetOpenFilename 方法多选文件时，返回的值是一个数组，该数组的下标也是 1。例如在以下代码中，数组变量 fileToOpen 的第一个值是 fileToOpen(1)。

```
Dim fileToOpen As Array = app.GetOpenFilename("图片文件(*.jpg), *.jpg", , "
请选择图片文件(支持 Ctrl+A 全选图片)", , True)
```

VS024：不再支持 Array 函数，而是改用{}来生成数组，并且可以生成二维数组。以下代码分别生成了一个一维数组和一个二维数组：

```
Dim Arr() = {"长江", "黄河", "松花江",  "珠江"}
Dim Arr2 = {{"广东", "东莞", "长安"}, {"四川", "成都", "陴县"}, {"湖北", "武汉","吴家山"}}
```

当然，如果一定要创建下标不为 0 的数组也可以，只是相当麻烦，代码如下：

```
'创建一个第一维下标为 0、上标为 10，第二维下标为 0、上标为 2 的一维数组
Dim Arr() As Integer = {10, 2}
'创建一个第一维下标为 0、上标为 2，第二维下标为 0、上标为 2 的一维数组
Dim Arr2() As Integer = {2, 2}
Dim Arr3 As Array
'赋值后 Arr3 则是二维数组，其第一维下标是 2、上标为 11，第二维下标为 2、上标为 3
Arr3 = Array.CreateInstance(GetType(String), Arr, Arr2)
```

17.3.2 新增功能

VSTO 对数组的管理比较强大，相对于 VBA 新增了很多新功能，此处简单介绍几个比较有用的功能。

<u>VS025</u>：增加了 Contains 属性，用于判断数组中是否包含某个值，代码如下：

```
Dim Arr() = {"长江", "黄河", "松花江",  "珠江"}
MsgBox(Arr.Contains("黄河") ) '返回值为 True，表示存在包含关系
```

<u>VS026</u>：增加了计算数组维数的属性 Rank，代码如下：

```
Dim aa(2, 4, 6) As Array
MsgBox(aa.Rank) '结果为 3
```

<u>VS0267</u>：增加了 Sort 方法，可以对数组进行升序排列，代码如下：

```
Dim Arr() As String = {10, 8, 2, "ABC"}
Array.Sort(Arr)
```

<u>VS028</u>：VSTO 中新增了 Array.Copyto、Array.Copy 和 Array.ConstrainedCopy，都可用于复制数组。第一个方法只对一维数组有效，后两种方法可以复制一维与二维数组。此处仅介绍 Array.Copyto。

```
Dim Arr() As String = {"A", "B", "C", "D", "E"}
Dim Arr2() As String = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
Arr.CopyTo(Arr2, 2) '将数组 Arr 的值复制到数组 Arr2 中，起始位置为第 3 个
```

复制后数组 Arr2 的值包含“1,2,A,B,C,D,E,8,9,10”。

17.4 窗体

相对来说，VSTO 在窗体方面的变化是最大的，首先是语法有变更，其次是新增了很多强大的新控件，另外对相同控件做了一些调整。本节主要介绍语法变更和功能调整。

17.4.1 名称变化

<u>VS029</u>：在 VSTO 中对窗体和几个常用控件的名称做了更改，主要体现在以下几个方面。

在 VBA 中窗体称为 UserForm1，而在 VSTO 中称为 Form1。

在 VBA 中按钮称为 CommandButton1，而在 VSTO 中称为 Button1。

在 VBA 中框架称为 Frame1，而在 VSTO 中称为 GroupBox1。

在 VBA 中多页控件称为 TabStrip1 或者 Page1，而在 VSTO 中称为 TabControl1。

<u>VS030</u>：相对于 VBA 中的窗体或者控件，VSTO 增加了很多新的事件，即使相同的事件在名称上也有所变化。

VBA 中的 UserForm_Initialize 事件，在 VSTO 中改成了 Form1_Invalidated 事件；

VBA 中的 UserForm_AddControl 事件，在 VSTO 中改成了 Form1_ControlAdded 事件；

VBA 中的 UserForm_Activate 事件，在 VSTO 中改成了 Form1_Activated 事件；

VBA 中的 UserForm_QueryClose 事件，在 VSTO 中改成了 Form1_Closed 事件；

VBA 中的 CommandButton1_DblClick 事件，在 VSTO 中改成了 Button1_DoubleClick 事件。

更多的事件名称变化不再一一举例，通常可以通过名称看出其功能。

有必要特别强调的是 VSTO 中增加了很多新的事件，从而让操作更加细化。

例如在 VBA 中只有一个关闭窗体的事件，而 VSTO 中则区分了 Closed 事件和 Closing 事件，先触发 Closing 事件，后触发 Closed 事件。

再如 VBA 中窗体或者按钮只有一个 Click 事件，而在 VSTO 中则同时有 Click 和 MouseClick 事件，前者在单击鼠标或者按【Enter】键时触发事件，后者只在单击鼠标时才触发事件。

还有，在 VBA 中对按钮控件提供了一个 MouseMove 事件，在鼠标移过按钮时触发，而在 VSTO 中则提供 4 个事件，包含光标进入按钮、移过按钮、停在按钮上方和离开按钮时触发，4 个事件分别为 MouseEnter、MouseMove、MouseHover 和 MouseLeave。

VS031：在 VBA 中可以直接使用 Userform1.show 代码显示窗体，而在 VSTO 中则必须通过 Dim 语句声明一个代表目标窗体的对象实例，然后通过这个变量调用窗体的 Show 方法，模板如下：

```
Sub Test()
  Dim f As New Form1
  f.Show()
End Sub
```

在显示窗体时最好加一句“f.TopMost = True”，表示让窗体置于所有窗口的最顶端。而在窗体中使用 Application.InputBox 或者 Application.FileDialog 调用“浏览”对话框时，则应该将窗体的 TopMost 属性修改为 False，避免“浏览”对话框被当前窗体覆盖，影响操作。

17.4.2 调用方式变化

VS032：VBAk 窗体和控件的 Caption 属性或者 Value 属性都被 VSTO 中的 Text 属性取代了。

VS033：VBA 中获得焦点的 SetFocus 方法在 VSTO 中修改成了 Focus 方法。

VS034：窗体控件的颜色不再是一个数值，而是一个对象。VSTO 提供了两个方法让窗体控件的颜色与代表单元格背景色的颜色值互相转换，FromOle 方法表示将颜色数值转换成窗体控件的颜色，ToOle 方法则相反。

```
Sub test()
  Dim Rng As Excel.Range= app.Range("A1") '声明变量并赋值
  '让标签控件的背景色等于单元格的背景色
  lable1.BackColor = System.Drawing.ColorTranslator.FromOle(Rng.Interior.Color)
  '让单元格的背景色等于标签控件的背景色
  Rng.Interior.Color = System.Drawing.ColorTranslator.ToOle(lable1.BackColor)
End Sub
```

VS035：要在 VBA 中设置窗体中的某个控件的提示信息，对该控件的 ControlTipText 属性赋值即可，而在 VSTO 中的步骤则是先在窗体中添加一个 ToolTip 控件，然后在窗体的 Load 事件中通过 SetToolTip 方法对控件指定提示信息。

例如要对窗体中的 Button1 控件添加提示信息，代码如下：

```
Private Sub Form2_Load(sender As Object, e As EventArgs) Handles MyBase.Load
```

```
    Me.ToolTip1.SetToolTip(Button1, "单击此按钮可执行 XXX 命令")
End Sub
```

第一参数是对象，第二参数是要显示的内容，代码运行效果如图 17-4 所示。

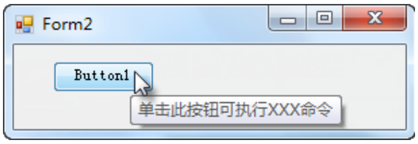


图 17–4　为控件添加提示信息

VS036：在判断用户在文本框输入过程中是否按下了某个键时，VSTO 的判断过程有了较大的变化。在 VSTO 中，文本框的 KeyPress 事件有 sender 和 e 两个参数，其中参数 e 即代表按键。可以将按键转换成字符，然后与内置的常量进行比较，从而判断用户按下了哪个键。

以下事件过程是在文本框中按下任意键时触发，事件过程可以判断用户是否按下了回车键或者录入了任意大写字母。

```
Private Sub TextBox1_KeyPress(sender As Object, e As System.Windows.Forms.KeyPressEventArgs) Handles TextBox1.KeyPress
        '判断用户是否按下了回车键，或者录入了大写字母
        If e.KeyChar = Chr(System.Windows.Forms.Keys.Return) Then MsgBox("你按下了回车键")
        If Asc(e.KeyChar.ToString) >= 65 And Asc(e.KeyChar.ToString) <= 90 Then MsgBox("你录入了大写字母")
End Sub
```

判断用户是否输入了数值则用以下代码：

```
If Not IsNumeric(e.KeyChar.ToString) Then MsgBox("你输入的不是数值" )
```

VS037：在 VSTO 中字体是一个类，需要创建一个 Font 类，然后将它赋值给 Font 对象。例如要对窗体中 Textbox1 的字体设置为黑体、12 号且加粗，那么代码如下：

```
Textbox1.Font = New System.Drawing.Font("黑体", 12,System. Drawing.FontStyle.Bold)
```

如果要将标签 Label1 的字体设置为倾斜，那么代码如下：

```
Label1.Font = New System.Drawing.Font(Me.Font, System. Drawing.FontStyle.Italic)
```

若要恢复不倾斜的状态，则用以下代码：

```
Label1.Font = New System.Drawing.Font(Me.Font, System.Drawing. FontStyle.Regular)
```

VS038：将窗体控件上移一层不再使用 Zorder 属性，而是采用 BringToFront 方法。

例如代码“ TextBox1.BringToFront()”可以让文本框上移一层。

VS039：关闭窗体的方法产生了变化。

VBA：Unload me

VSTO：Me.Close()

VS040：列表框是否允许多选的控制方式产生了变化，例如让列表框支持多选，代码如下。

VBA：ListBox1.MultiSelect = fmMultiSelectMulti

VSTO：ListBox1.SelectionMode = System.Windows.Forms.SelectionMode.MultiExtended

VS041：列表框中添加单个值的代码有所变化。

在 VSTO 中添加了 Items 属性，有很多操作都需要借助 Items 来实现，即 VBA 中的部分方法和属性移到了 Items 的下面，需要通过 Items 才能调用。

VBA：ListBox1.AddItem Item

VSTO：ListBox1.Items.Add(Item)

VS042：列表框中添加一组值的代码有所变化。

VBA：

```
ListBox1.list=Arr     'Arr 是一维数组
```

VSTO：

```
ListBox1.Items.AddRange(Arr)
```

VBA 中“ListBox1.list=Arr”这种赋值方式会用新值覆盖原值（假设列表框中已经有值），而 VSTO 中的 AddRange 方法属于追加数据，不会覆盖原值，因此如果要达到 VBA 的相同功能，应该在赋值前加一句“ListBox1.Items.Clear()”。

VS043：删除列表框中的当前选中的条目，VSTO 的语法稍有变化。

VBA：Me.ListBox1.RemoveItem Me.ListBox1.ListIndex

VSTO：ListBox1.Items.Remove(ListBox1.Text)

VS044：VSTO 中取消了列表框的 List 属性，以下代码展示了在引用列表框的单个值时，二者的不同之处。

VBA：ListBox1.List(1)

VSTO：ListBox1.Items(1)

在引用列表框的所有值时，VBA 可以用以下这句代码完成。

```
Range("a1").Resize(ListBox1.ListCount, 1) = ListBox1.List
```

而 VSTO 则麻烦得多，必须借助 CopyTo 方法，将列表框的值赋值给数组，然后再将数组的值写入到区域中。

```
Dim Arr(ListBox1.Items.Count - 1) As Object
ListBox1.Items.CopyTo(Arr, 0)
app.Cells(1).Resize(ListBox1.Items.Count, 1).value = app.WorksheetFunction.
Transpose(Arr)
```

同理，对于某些非花扩号生成的，下标不为 0 的数组，要将数组赋值给列表框也不能直接赋值，需要先利用 CopyTo 方法转到数组变量中，再将数组变量的值赋予列表框。例如：

```
Dim ArrList
ArrList= app.GetOpenFilename("请选择文件(*.*）,*.*", , , , True)
If TypeName(ArrList) = "Object()" Then
    Dim Arr(ArrList.LongLength - 1) As Object        '其中 LongLength 代表数组的元素个数
    FileArr.CopyTo(Arr, 0)
    ListBox1.Items.AddRange(Arr)                     '将用户选择的所有文件名称导入到列表框中
End If
```

VS045：VSTO 中对列表框的项目计数方式有所变化。

VBA：ListBox1.ListCount

VB.net：ListBox1.Items.Count

VS046：让列表框的某一行呈选中状态的方法由 Selected 改成了 SetSelected，而且语法也不相同，以“选中第三行”为例。

VBA：ListBox1.Selected(2) = True

VSTO：ListBox1.SetSelected(2, True)

以上关于列表框的所有变化内容，同样适用于组合框控件。

17.4.3 功能变化

VS047：在 VSTO 中，Height 和 Width 属性是以像素为单位的；而在 VBA 中，窗体的 Height 和 Width 属性则以磅为单位。磅与像素的换算单位是：1 磅等于 4/3 像素，约 1.333333 像素。因此，以下两句代码获得的窗体高度基本一致。

VBA：Me.Height = 400

VSTO：Me.Height = 400 * 1.333333

VS048：VBA 中的列表框可以显示多列，可以将二维数组赋值给列表框，而 VSTO 中不允许显示多列，只支持一维数组。

如果要在窗体中显示二维数组，那么可以用 ListView 控件代替列表框，不过 ListView 控件最大的缺点是不能像 VBA 中的列表框那样一句话赋值一个二维数组给控件，而是需要一个一个将值添加进去，在效率上较差，代码也相当长。

17.5　字典与正则表达式

关于字典与正则表达式，VSTO 提供了新的调用方式，但是同时也保留了与 VBA 相同的语法，兼容 VBA 规范的代码。

17.5.1　字典

VS049：VBA 的代码可以直接应用于 VSTO 中，只是在将字典的值输出到单元格时要对 Range 对象添加.Value，其他都一样。

```
Sub 字典应用模板()
  With CreateObject("scripting.dictionary")
      .Add("1", "A")
      .Add("2", "B")
      .Add("3", "C")
      app.Range("a1").Resize(1, .Count).Value = .keys
      app.Range("a2").Resize(1, .Count).Value = .items
  End With
End Sub
```

以下是 VSTO 新增的字典用法：

```
Sub 字典应用模板升级版()
    Dim Dic As New Dictionary(Of String, String)
    With Dic
        .Add("1", "A")
        .Add("2", "B")
        .Add("3", "C")
        APP.Range("A1").Resize(1, .Count).Value = .Keys.ToArray()
        APP.Range("A2").Resize(1, .Count).Value = .Values.ToArray()
    End With
End Sub
```

17.5.2　正则表达式

VS050：VSTO 完全兼容 VBA 中的正则表达式语法，不过同时也新增了自己的语法，模板如下：

```
Sub 字典基本用法模板()
  Dic Dic As New Dictionary(Of String, String)
  With Dic
     .Add("1", "A")
     .Add("2", "B")
     .Add("3", "C")
     APP.Range("A1").Resize(1, .Count).Value = .Keys.ToArray()
```

```
        APP.Range("A2").Resize(1, .Count).Value = .Values.ToArray()
    End With
End Sub
```

17.6 菜单与功能区

Excel 2016 支持传统的工具栏和工作表菜单，但是传统的工具栏和工作表菜单已经边缘化了，今后将会大力推广功能区菜单。但是本节仍然阐述在 VSTO 中为 Excel 插件创建传统菜单和功能区按钮的代码与 VBA 之间的差异。

17.6.1 工作表菜单

VS051：在 VSTO 中创建工作表菜单采用以下模板即可：

```
WithEvents 子菜单 As Office.CommandBarButton
Public Sub 建菜单()
子菜单 = app.CommandBars(1).Controls.Add(Microsoft.Office. Core.
MsoControlType.msoControlButton, , , , True)
    With 子菜单
        .Style = Microsoft.Office.Core.MsoButtonStyle.msoButtonIconAndCaption
        .Caption = "新菜单"
        .FaceId = 483
    End With
End Sub
Private Sub 子菜单_Click(Ctrl As Microsoft.Office.Core.CommandBarButton, ByRef CancelDefault As
Boolean) Handles 子菜单.Click
    MsgBox("玩转 VSTO")
End Sub
```

使用以上模板的方法是：需要在创建菜单时，将以上代码复制到模块中，然后根据需求修改菜单的 Caption 和 FaceId 两个参数，再将要关联的过程放到“子菜单_Click”事件中，最后在“ThisAddIn_Startup”事件加上代码“Call 建菜单()”即可，代码运行效果如图 17-5 所示。调用创建菜单的方法在第 19 章中有应用案例。

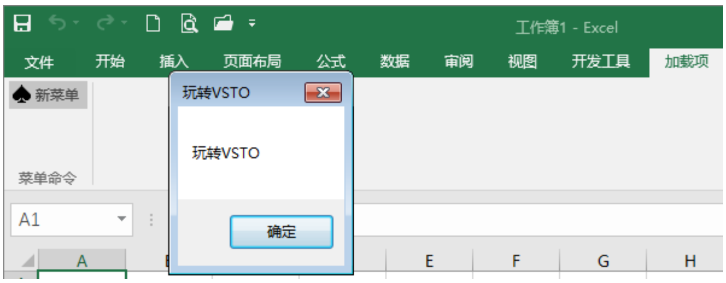


图 17–5　菜单外观

17.6.2 功能区菜单

功能区按钮的设计过程在前 16 章已经讲过，本章不再讲述 VBA 与 VSTO 设计功能区按钮的差异，而是讲解回调方面的差异。

VS052：在 VSTO 中为功能区按钮绑定 Sub 过程比 VBA 中更方便，具体方法是双击按钮从而进入功能 Ribbon 控件的代码窗口，此时光标会自动进入当前按钮的 Click 事件过程中，在此

处直接录入事件过程的代码即可。如果在模块中已经录入了 Sub 过程，那么在此处调用该过程名称即可。

VS053：在 VBA 中使用回调控制功能区按钮的 Label 和图像是比较困难的，步骤相当烦琐，而在 VSTO 中只要一句代码即可回调。假设已经设置好了一个功能区按钮，其 Label 属性的默认值为“Yes”，其 OfficeImageId 属性的默认值为“AcceptInvitation”，要求单击按钮后将 Label 属性的值改为“No”，OfficeImageId 属性值改为“DeclineInvitation”，再次单击按钮返回默认值，那么可用以下代码完成：

```
    Imports Microsoft.Office.Tools.Ribbon '此句必须放在最顶端，用于声明命名空间
    Private Sub Button1_Click(sender As Object, e As RibbonControlEventArgs) Handles Button1.Click '此代码
'放在 Class 与 End Class 之间
        Button1.Label = IIf(Button1.Label = "Yes", "No", "Yes")
        Button1.OfficeImageId = IIf(Button1.OfficeImageId = "DeclineInvitation", "AcceptInvitation",
"DeclineInvitation")
    End Sub
```

图 17-6 和图 17-7 分别是按钮的默认状态和单击按钮后的状态。在图 17-7 状态下再次单击按钮，按钮会回到默认状态。

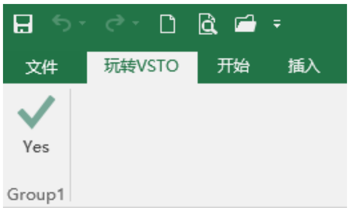


图 17–6　默认状态

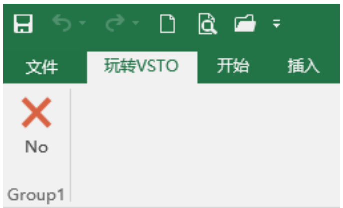


图 17–7　单击后的状态

VS054：上一段代码只能在 Ribbon 组件的代码窗口中使用，如果在模块的 Sub 过程中实现回调就需要用不同的思路了。

假设仍然要实现上面的需求，但需要在模块的 Sub 过程中完成，那么代码如下：

```
    Sub abc()
        Dim Btn1 As Microsoft.Office.Tools.Ribbon.RibbonButton =
Globals.Ribbons.Ribbon1.Tab1.Groups(0).Items(0)
        Btn1.Label = IIf(Btn1.Label = "Yes", "No", "Yes")
        Btn1.OfficeImageId = IIf(Btn1.OfficeImageId = "DeclineInvitation", "AcceptInvitation", "DeclineInvitation")
    End Sub
```

其中 Groups(0)代表第一组，Items(0)代表第一组中的第一个按钮。

当然也有更简单的办法，不借助变量，添加前缀“Globals.Ribbons.Ribbon1”后直接引用 Button1，代码如下：

```
    Sub abc()
     With Globals.Ribbons.Ribbon1.Button1
      .Label = IIf(.Label = "Yes", "No", "Yes")
      .OfficeImageId = IIf(.OfficeImageId = "DeclineInvitation", "AcceptInvitation", "DeclineInvitation")
     End With
    End Sub
```

17.7　管理文件与目录

管理文件包含文本文件的读取与写入，以及复制文件、移动文件、删除文件。而管理目录则

包含新建文件夹、删除文件夹、移动文件夹和重命名文件夹。VSTO 与 VBA 的命令名称是基本一致的，二者仅仅调用方式不同。

17.7.1 管理文件

VS055：VSTO 中保留了 VBA 的处理文本文件的语法，同时也新增了自己专用的语法。以下代码表示在 C 盘新建一个名为 test.bat 的文本文件，然后向该文件写入数据，写完后关闭文件，最后执行这个文件中的命令。命令的功能是获取当前计算机的 IP 地址。

```
Sub 创建文本文件且执行文件的模板()
    Dim SW As IO.StreamWriter = My.Computer.FileSystem.OpenTextFileWriter("c:\test.bat", False, System.Text.Encoding.GetEncoding("GB2312"))
    SW.WriteLine("@echo off" )   '写入第一行
    SW.WriteLine("ipconfig" )   '写入第二行
    SW.WriteLine("pause")
    SW.Close()
    System.Diagnostics.Process.Start("c:\test.bat")
End Sub
```

VS056：在读取文本文件的所有内容方面，VSTO 中新增了比 VBA 更简单的代码。

假设要读取 C 盘中 123.txt 文件的内容，使用以下模板即可：

```
Sub 读取文本文件模板()
    Dim sr As New System.IO.StreamReader("c:\123.txt", Encoding.Default)
    MsgBox(sr.ReadToEnd())
    sr.Close()
End Sub
```

VS057：在 VBA 中用 CreateObject("Scripting.FileSystemObject")方法创建 FSO 对象的引用，然后通过它调用 FSO 的方法和属性；在 VSTO 中则用 My.Computer.FileSystem 调用 FSO 的所有方法和属性。VSTO 与 VBA 中的 FSO 对象调用方式不同，语法也有一定差异。

复制文件：My.Computer.FileSystem.CopyFile(源文件名,目标文件名，是否覆盖文件)

命名文件：My.Computer.FileSystem.RenameFile(要重命名的文件名称,新名称)

需要注意的是，RenameFile 的第二参数不能带路径，这点与 VBA 不同。

命名文件或文件夹：My.Computer.FileSystem.Rename(原本的路径,新路径)。

需要注意的是，Rename 方法既能命名文件也能命名文件夹，它的第二参数必须带有路径。

删除文件：My.Computer.FileSystem.DeleteFile(文件路径)

判断文件是否存在：My.Computer.FileSystem.FileExists(文件路径)

VS058：获得文件的创建时间、最后一次访问时间和修改时间。

```
Dim Finfo As IO.FileInfo = My.Computer.FileSystem.GetFileInfo("C:\123.txt")
MsgBox(Finfo.CreationTime)        '创建时间
MsgBox(Finfo.LastAccessTime)      '最后一次访问时间
Msgbox(Finfo.LastWriteTime)       '最后一次修改时间
```

VS059：获取文件的扩展名，使用 Extension 属性即可。

```
Dim Finfo As IO.FileInfo = My.Computer.FileSystem.GetFileInfo("D:\报表.docx")
MsgBox(Finfo.Extension)
```

17.7.2 管理目录

VS060：VSTO 用于管理目录的代码与 VBA 相比稍有变化。

命名文件夹：My.Computer.FileSystem.RenameDirectory(文件夹名称,新名称)

新建文件夹：My.Computer.FileSystem.CreateDirectory(文件夹路径)

删除文件夹：My.Computer.FileSystem.DeleteDirectory(路径,文件夹非空时的处理方式)

如果要删除的文件夹中有文件，那么对第二参数赋值为 FileIO.DeleteDirectoryOption.DeleteAllContents 时表示删除所有文件和目录，在对第二参数赋值为 FileIO.DeleteDirectoryOption.ThrowIfDirectoryNonEmpt 时则表示不删除文件和目录，但会产生一个异常（异常不等于错误，可以使用 Try...End Try 语句捕获这个异常）。

移动文件夹：My.Computer.FileSystem.MoveDirectory(原路径,新路径)

VS061：在 VBA 中要获得桌面、我的文档等路径需要调用脚本语言，而在 VSTO 中调用内置语言即可。

VBA：

CreateObject("Wscript.Shell").SpecialFolders.Item("Desktop")——桌面

CreateObject("Wscript.Shell").SpecialFolders.Item("Favorites")——收藏夹

CreateObject("Wscript.Shell").SpecialFolders.Item("AllUsersStartMenu")——开始菜单

CreateObject("Wscript.Shell").SpecialFolders.Item("MyDocuments")——我的文档

VSTO：

My.Computer.FileSystem.SpecialDirectories.Desktop——桌面

Environment.GetFolderPath(Environment.SpecialFolder.Favorites)——收藏夹

Environment.GetFolderPath(Environment.SpecialFolder.StartMenu)——开始菜单

My.Computer.FileSystem.SpecialDirectories.MyDocuments ——我的文档

17.8　杂项

本节讲述若干在开发插件时比较有用的小功能。

VS062：在 VSTO 中关于在剪贴板中写入数据或读取数据方面提供了比 VBA 更简化的代码。

假设需要将 A1 和 A2 的值串成一个字符串写入到剪贴板中，使用 VBA 完成需要先引用 FM20.dll 文件，然后在模块中录入以下代码：

```
Sub VBA 写入剪贴板()
Dim MyData As DataObject                      '声明变量
Set MyData = New DataObject                   '新建一个 DataObject 类的实例
MyData.SetText Range("A1") & Range("A2")      '将 A1 与 A2 的值合并，然后赋值给变量
MyData.PutInClipboard                         '将变量的值写入到剪贴板中
End Sub
```

而在 VSTO 中仅需一句代码即可完成：

```
Sub VSTO 写入剪贴板()                          '将 A1 与 A2 的值合并,然后写入剪贴板
Windows.Forms.Clipboard.SetText(app.Range("A1").Value & app.Range("A2").
Value)
End Sub
```

VS063：VSTO 提供了播放 wav 音乐的办法，仅需一句代码：

```
My.Computer.Audio.Play("c:\ 凡人歌.wav")
```

如果播放 MP3 格式的音乐则需要调用 API 函数了，代码如下：

```
Private Declare Function mciSendString Lib "winmm.dll" Alias "mciSendStringA" (ByVal lpstrCommand As
String, ByVal lpstrRetumString As String, ByVal uReturnLength As Int16, ByVal hwndCallback As Int16) As Int16
Sub 播放 Mp3()
```

```
        mciSendString("open C:\凡人歌.mp3" & " alias mp3", 0, 0, 0)
        mciSendString("Play mp3", 0, 0, 0)
    End Sub
    Sub 关闭()
        mciSendString("close mp3", 0, 0, 0)
    End Sub
```

在 VBA 中也可以使用以上代码，但是 Int16 需要改成 Long 或者 Integer，VBA 中没有 Int16 这个数据类型。

VS064：判断当前是否按下了 Shift/Ctrl 键，VBA 中未提供此类工具，调用 API 函数可以实现。VSTO 中一句代码即可实现。

```
If My.Computer.Keyboard.ShiftKeyDown Then MsgBox("您按下了 Shift 键")
If My.Computer.Keyboard.CtrlKeyDown Then MsgBox("您按下了 Ctrl 键")
```

VS065：VSTO 中不再支持 Ontime 和 OnKey 方法，但是对于 Sendkeys 还是支持的，只是调用方式发生了改变，不再通过 Application 方式调用，而是用 My.Computer.Keyboard 方式调用。

```
My.Computer.Keyboard.SendKeys("%ti")        '发送 Alt+T+I 快捷键，打开“加载宏”对话框
```

VS066：在 VSTO 中读写注册表使用 SaveSetting 和 GetSetting 方法，它们只能操作固定键值的注册表信息；在 VSTO 中使用 SetValue 和 GetValue 方法，可以读写任意键值。

以下代码表示用 SetValue 在“HKEY_CURRENT_USER\SOFTWARE\Microsoft\ABCD”这个路径下创建一个名为“EFG”的子键，且指定键值为 123。然后使用 GetValue 读取该键值：

```
My.Computer.Registry.SetValue("HKEY_CURRENT_USER\SOFTWARE\Microsoft\ABCD", "EFG", 123,
Microsoft.Win32.RegistryValueKind.DWord)
MsgBox(My.Computer. Registry.GetValue("HKEY_CURRENT_USER\SOFTWARE\Microsoft\ABCD", "EFG", 0))
```

VS067：在 VBA 中未提供播放 gif 动画的代码，在 VSTO 中可以借助窗体控件 PictureBox 来播放动画，方法是在窗体中添加一个 PictureBox1 控件，然后在窗体的 Form1_Load 事件中加入以下代码：

```
PictureBox1.Image = Image.FromFile("c:\123.gif")   '请根据实际情况修改路径
```

VS068：在 VBA 中代码错误分为编译错误和运行时错误，有些编译错误会在运行前通过颜色提示用户，还有些编译错误在运行时弹出对话框通知用户。而运行时错误则是在运行代码时弹出对话框通知用户。在 VSTO 中不仅有错误，还有异常，代码产生错误时会明显地通知用户，但是当代码产生异常时则可能不产生任何通知，同时也不产生正确结果，此时想要解决问题就比较困难。

VSTO 提供了捕获异常的工具 Try…End Try，可通过它找出所有异常。以下代码的功能是让用户在输入框中录入年龄，然后利用 Int 函数将年龄取整，再通过 Msgbox 展示在对话框中。

```
Sub test()
  Dim myval As Object = app.InputBox("请输入年龄")
  MsgBox("您今年：" & Int(myval) & "岁整")          '使用 Int 表示当年岁有小数时，只取整数
End Sub
```

当把代码封装在 DLL 文件中，然后将 DLL 文件加载到 Excel 中，执行以上代码并且输入中文后，程序不会弹出任何错误提示，但也不会得到需要的结果。如果在 VBA 中执行以上代码（需要稍加修改），且在输入框中输入中文，程序一定会弹出“类型不匹配”的错误提示。在 VSTO 中必须使用 Try…End Try 才能捕捉到异常，从而生成提示信息。

```
Sub test()
  Try
      Dim myval As Object = app.InputBox("请输入年岁")
      MsgBox("您今年：" & Int(myval) & "岁整")      '使用 Int 表示当年岁有小数时，只取整数
```

```
        Catch ex As Exception
            MsgBox(ex.ToString)
        End Try
    End Sub
```

将代码改成以上形式后，如果用户输入正常的值，那么程序的执行结果和不采用 Try...End Try 语句时完全一样；如果用户录入了文本，那么执行过程中会弹出异常提示，包含异常的原因和产生异常的模块名称、过程名称，以及代码的行数。

VS069：Excel 的 VBA 代码可以直接按【F8】键逐句运行从而调试代码，利用 VSTO 开发 Excel 的外接程序时无法调用【F8】键，要逐句运行代码可以按以下步骤执行。

选中需要调试过程的第一句，然后按下【F9】键，表示在此设置一个断点。接着单击工具栏的"启动"按钮或者按【F5】键启动插件，当打开 Excel 程序后，单击功能区按钮运行 Sub 过程，此时会激活 Visual Studio 2015 的代码窗口，同时 Sub 过程会执行到设置断点的位置并自动停下。此时每按一次【F8】键，Sub 过程会执行一句代码，从而便于用户调试代码。

在调用过程中，将鼠标指针拖曳至变量时，在屏幕上会显示一个对话框，对话框中包含变量名称和变量的当前值。

17.9　课后思考

1．在 VBA 中的 Integer、Double 与 VSTO 中的 Integer、Double 有何差异?

2．在 VSTO 中写入注册表的操作比 VBA 中写入注册表的操作有何优势?

3．在以下代码中，Arr 和 Arr2 两个变量分别是什么数据类型?

```
Dim Arr, Mystr As String, Arr2(), FileCount As Int16
```

4．先打开 Excel，后打开 Word，有没有可能让 Excel 中的窗体显示在 Word 窗体上方?

5．窗体中有一个 Label，它的内容是一个网址。应该如何编写代码才能让鼠标指针移过标签时标签的文字显示为倾斜状态，而鼠标指针离开标签时标签的文字恢复正常状态?

第 18 章　将 VBA 插件升级为 VSTO 插件

学习 VSTO 有两种方法，一是直接学习 VSTO 代码，二是在使用 VBA 的前提下，掌握 VBA 与 VSTO 的语法差异，然后直接将 VBA 代码升级为 VSTO 代码，此方法对于 VBA 熟练者而言可以更快地学会 VSTO，学习进度提升数倍。

本书第 17 章展示了 VBA 与 VSTO 的语法差异，利用这些知识足以将 VBA 插件升级为 VSTO 插件，从而使插件可同时用于 32 位和 64 位的 Excel，以及 WPS。

本章以具有代表性的 3 个工具为例展示 VBA 代码的升级过程，3 个过程包含“创建工资条”、“文件批量命名”和“零值控制器”，其中“创建工资条”属于普通的 Sub 过程，“文件批量命名”属于窗体，“零值控制器”则属于类和事件过程，它们分别代表了 3 类最常见的应用。

18.1　设计插件框架

通过 VB.net 开发 Excel 插件前需要做一些准备工作，包含设置外插程序的 Office 版本、.net framework 的版本、代码的保存位置、插件的名称、版本编号、插件的输出路径、外部引用、签名和菜单，当然其中部分项目可以采用默认设置。

18.1.1　VBA 插件介绍

在升级 VBA 插件前，有必要了解一下 VBA 插件的菜单、功能，以及使用方式。

VBA 插件的存放位置如下（第 2 个文件用于测试插件功能）:

本例案例文件请参考：..\ 第 18 章\ Excel 精灵.xlam

使用第 14 章的知识安装以上插件，然后打开 Excel，可以看到菜单布局方式如图 18-1 所示。

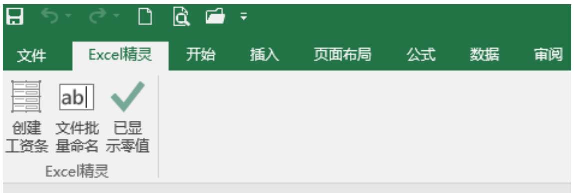


图 18–1　插件的菜单布局

其中“创建工资条”工具的功能是将工资明细表转换成工资条形式的表格，从而便于裁剪；“文件批量命名”工具的功能是对指定的文件批量命名，包含在原文件名前面插入指定字符、在原文件名后面插入指定字符，以及批量替换文件名中指定的字符；“零值控制器”工具的功能是

控制单元格中零值的显示状态，单击按钮后隐藏零值，再次单击则显示零值。

此处不再一一演示三个工具的操作方法。笔者录制了一个操作视频，和本书的案例文件保存在一起，读者可以从案例文件中找到它，具体位置是“..\ 第 18 章\ 工具演示.mp4”。

18.1.2　设计插件框架

利用 VSTO 开发 Excel 外接程序，相对于 VBA 开发插件既有优势也有劣势，优势体现在编写代码的过程，劣势体现在编写代码前必须做不少设置工作以及编写后的封装工作，否则他人无法安装插件。

在编写代码前，需要做的准备工作如下：

1．打开 Visual Studio 2015，在起始页中单击“新建项目”。

2．在“新建项目”对话框中选择“Excel 2013 和 2016VSTO 外接程序”，将上方的.Net Framework 版本号由 4.5.2 修改为 4，并将名称改为“Excel 精灵”，然后单击“确定”按钮。如果使用 Excel 2010 版本，则可以选择“Excel 2010 外接程序”，后续步骤一样，插件同样可以用于 Excel 2010、Excel 2013 或者 Excel 2016 中。

3．选择菜单“项目”→“Excel 精灵属性”，然后进入“应用程序”选项卡，单击“程序集信息”按钮，在弹出的“程序集信息”对话框中可以看到当前插件的标题和版本号，此时可以随意修改，例如“Excel 精灵 VIP 版”，程序集版本为“2.2.0.0”（只能录入 4 个数值，不需要输入点号）。还可以在此对话框中指定说明、公式、商标等信息。

4．选择菜单“项目”→“添加组件”，选择最上方的“功能区(可视化设计器)”，然后单击“添加”按钮。

5．在 Ribbon1 右方或者下方的空白区域单击右键，选择“删除”命令，从而删除控件自带的 Tab 组件。然后进入工具箱，将 Tab 控件拖到 Ribbon1。如果没有看到工具箱，那么可以选择菜单“视图”→“工具箱”，从而激活工具箱。

6．打开右下角“属性”对话框，将 Tab1 对象的 Label 属性值由“Tab1”修改为“Excel 精灵”。

7．单击“Position”旁边的田字标展开子选项，将“PositionType”属性修改为“BeforeOfficeId”，然后将下面的“OfficeId”修改为“TabHome”，表示新插入的选项卡显示在“开始”选项卡的左方。

8．进入工具箱，将 Group 组件拖到上一步所建立的新选项卡中，然后打开“属性”对话框，将 Label 属性的默认值“Group1”修改为“Excel 精灵”。

9．进入工具箱，将 Button 组件拖曳到上一步所建立的 Group 中，然后打开“属性”对话框，将其 Label 的值修改为“创建工资条”，将 OfficeImageId 的值修改为“ViewsLayoutView”，将 ControlSize 的值修改为“RibbonControlSizeLarge”。

10．重复第 9 步，继续添加两个按钮，将第一个按钮的 Label 的值修改为“文件批量命名”，图标名称设置为“FormControlEditBox”；将第二个按钮的 Label 的值修改为“已显示零值”，图标名称设置为“AcceptInvitation”，此时功能区菜单的效果如图 18-2 所示。

此时尽管只有菜单还没有写任何代码，但是按下【F5】键后已经可以在 Excel 中看到插件了，效果如图 18-3 所示。

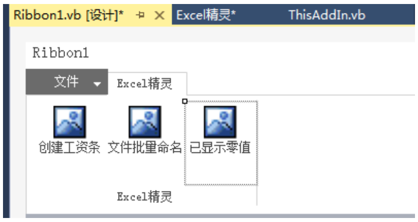


图 18–2　功能区菜单布局

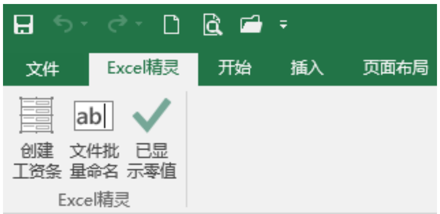


图 18–3　插件在 Excel 中的显示效果

18.2　升级 Sub 过程“创建工资条”

在 VSTO 中设计好“Excel 精灵”插件的大框架后，本节重点演示升级 VBA 插件第一个菜单“创建工资条”的代码，使其在 VSTO 插件中也能顺利运行。

18.2.1　准备工作

由于 Excel 的外接程序需要调用 Excel 的对象，因此需要声明一个代表 Excel 对象的公共变量，然后把 VBA 代码复制到 VSTO 的代码窗口中，并稍加修改使其符合 VB.net 语法。具体操作步骤如下。

1．选择菜单“项目”→“添加模块”，然后单击“添加”按钮，从而生成“Module1.vb”。

2．在模块中“Module Module1”的下一行录入以下代码：

```
Public app As Excel.Application = Globals.ThisAddIn.Application
```

3．在代码窗口上方的导航栏中选择“Ribbon1[设计]”，然后双击第一个菜单“创建工资条”，此时程序会进入 Ribbon 控件的代码窗口中，而且会自动生成 Button1 控件的 Click 事件过程外壳，代码如下：

```
Private Sub Button1_Click(sender As Object, e As RibbonControlEventArgs)
Handles Button1.Click

End Sub
```

4．在 VBA 插件中找到过程“创建工资条”，将它的代码从 Excel 中复制到以上 Click 事件过程中（只复制代码，忽略程序外壳）。

从图 18-4 中可以看到，把 VBA 代码复制到 VB.net 的代码窗口中之后，在代码下方会产生大量的波浪线，表明这些波浪线所标示的地方代码有问题，不符合 VSTO 的语法，需要修改。

接下来，在下一节中将一一说明每个错误的修改方式。为了便于描述，特意对需要修改的代码行添加了编号，通过编号来说明当前要修改的代码的位置。



图 18–4　错误代码提示

18.2.2　修改代码

在第 17 章中介绍了 VBA 与 VSTO 的语法差异，根据第 17 章的内容足以将 VBA 代码修改为 VSTO 可用的代码。针对图 18-4 所示的错误代码，修改方法如下：

错误编码：1

修改结果：

```
app.ScreenUpdating = False
```

修改依据：VS004

错误编码：2

修改结果：

```
app.ActiveSheet.Copy(, app.Sheets(app.Sheets.Count))
```

修改依据：VS004 和 VS014

错误编码：3

修改结果：

```
For Item = app.ActiveSheet.UsedRange.Rows.Count To 3 Step -1
```

修改依据：VS004

错误编码：4

修改结果：

```
app.Cells(Item, 1).Resize(2, 1).EntireRow.Insert
```

修改依据：VS004

错误编码：5

修改结果：

```
RowNum = app.ActiveSheet.UsedRange.Rows.Count
```

修改依据：VS004

错误编码：6

修改结果：

```
app.Rows("1:1").Copy
```

修改依据：VS004

错误编码：7

修改结果：

```
Dim Rng As Excel.Range
```

修改依据：VS004

错误编码：8

修改结果：

```
Rng = app.Rows("4:4")
```

修改依据：VS009 和 VS004

错误编码：9

修改结果：

```
Rng = app.Union(Rng, app.Rows(Item))
```

修改依据：VS009 和 VS004

错误编码：10

修改结果：

```
app.ActiveSheet.Paste
```

修改依据：VS004

错误编码：11

修改结果：

```
.Offset(-1, 0).Borders.LineStyle = Microsoft.Office.Core.XlConstants.xlNone
```

修改依据： VS012

错误编码：12

修改结果：

```
app.Range("a1").Select
```

修改依据：VS004

错误编码：13

修改结果：

```
app.ScreenUpdating = True
```

修改依据： VS004

18.3 升级窗体“文件批量命名”

升级窗体较之升级 Sub 过程的代码要复杂得多，主要体现在两个方面，其一是在 VBA 中的控件与 VSTO 中的控件名称不同，事件过程的名称也不同；其二是同一个控件在 VBA 中和 VSTO 中的功能会有所差异，属性和方法也有差异，例如 VBA 的列表框有 List 属性，而 VSTO 取消了该属性，这导致修改代码时需要耗费更多的精力。

VSTO 不能直接导入 VBA 窗体，因此本节需要先在 VSTO 中绘制一个同样的窗体，然后将 VBA 代码复制进来再做修改。

18.3.1　准备工作

插件中的“批量重命名”窗体比较复杂，用到了标签控件、命令按钮、文本框控件、列表框控件、框架控件和选项按钮。在 Visual Basic 6.0 版本中可以直接导入 VBA 窗体，但是 VB.net 不支持，因此需要在 VB.net 中手动绘制窗体。

绘制窗体比较简单，具体过程不再一一展示，通常选择菜单“项目”→“添加 windows 窗体”从而生成一个窗体，然后进入工具箱将需要的控件拖曳到窗体中，按正确的位置摆放即可，最终效果如图 18-5 所示。

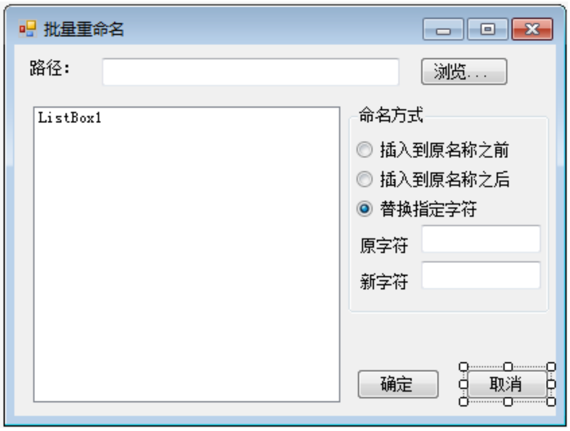


图 18–5　在 VB.net 中生成窗体

重点在于事件过程与控件的对应关系，因为 VBA 中的窗体与控件和 VSTO 中的窗体与控件在名称上不同，事件过程的书写方式也不同。为了尽可能减少修改代码的工作量，需要让两者的控件保持相同名称，操作方式如下：

1．选中窗体 Form1，打开右下角的“属性”窗口，将 Name 属性的默认值“Form1”修改为“UserForm1”。

2.选择窗体中的“Button1”，将其 Name 属性的默认值“Button1”修改为“CommandButton1”。然后采用相同办法修改其他两个命令按钮，要注意编号顺序与 VBA 保持一致，否则后面修改时会出错。

3．选择窗体中的“RadioButton1”，将其 Name 属性的默认值“Radio Button1”修改为“OptionButton1”，然后用相同办法修改其他两个选项按钮的 Name 属性值。

文本框控件和列表框控件的名称没有差异，因此不需要修改。

4．修改好窗体与控件的名称后，接下来应该复制 VBA 代码到 VSTO 的窗体中。此操作有两种思路，一种是一次性将窗体中所有代码复制过来，然后修改事件过程的名称；另一种是先在窗体的代码窗口上方选择对象名称和事件名称，从而生成事件过程的外壳，然后从 VBA 窗体中逐一复制每个事件过程的代码。本书中采用第二种办法，在上方的导航栏中选择对象名称和过程名称从而生成事件过程的外壳，最终结果如图 18-6 所示。

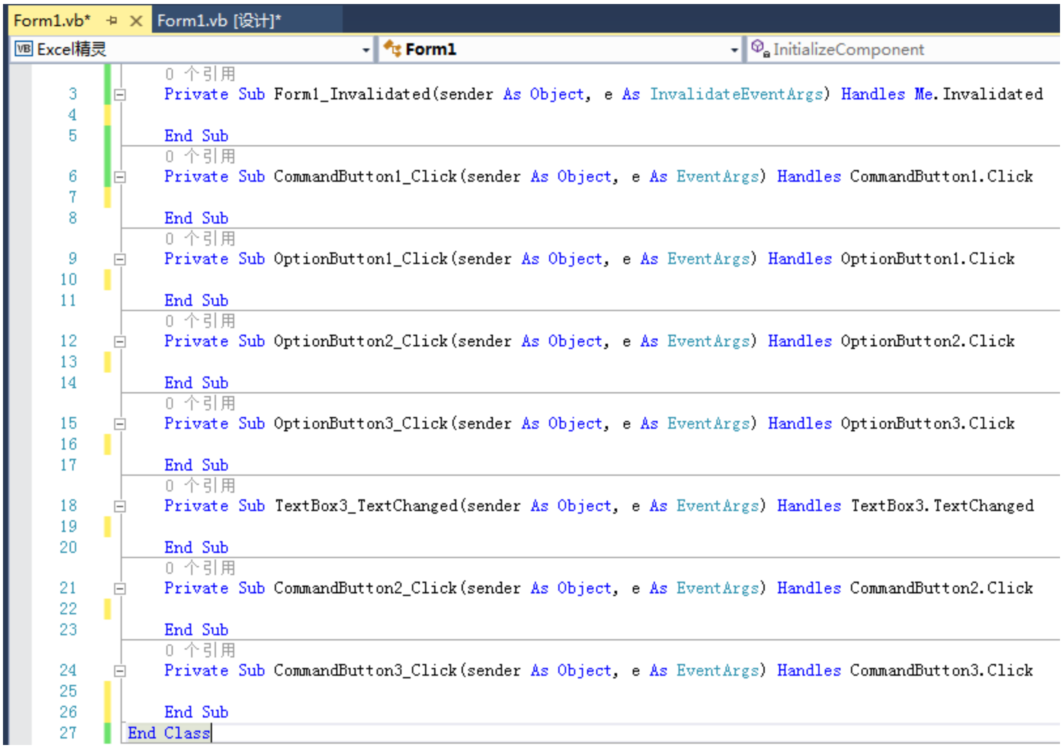


图 18–6　预先生成的事件过程外壳

5．把 VBA 窗体中对应的事件过程代码复制到 VSTO 中，效果如图 18-7 所示。

6．选中工具箱旁边的解决方案资源管理器，并双击其中的“Ribbon1.vb”从而进入功能区设计界面，然后双击功能区　“文件批量命名”按钮，此时程序会进入功能区的代码窗口，同时生成一个名为 Button2_Click 的事件过程外壳。



```
Form1.vb [设计]
CommandButton3    Click
1 个引用
Public Class Form1
    0 个引用
    Private Sub Form1_Invalidated(sender As Object, e As InvalidateEventArgs) Handles Me.Invalidated
        CommandButton2.Enabled = False '禁用命令按钮CommandButton2
    End Sub
    0 个引用
    Private Sub CommandButton1_Click(sender As Object, e As EventArgs) Handles CommandButton1.Click
        Dim Filearr  '声明一个变体型变量
        '弹出一个选择文件的对话框
        Filearr = Application.GetOpenFilename("所有文件(*.*),*.*", , , , True)
        If IsArray(Filearr) Then  '如果用户选择了文件(此时变量Filearr是数组)
            ListBox1.List = Filearr  '将选择的所有文件名称导入到列表框中
            TextBox1.Value = Replace(Filearr(1), Dir(Filearr(1)), "")
        Else  '否则
            Exit Sub  '否则结束过程
        End If
        CommandButton2.Enabled = True  '让CommandButton2呈可用状态
    End Sub
    0 个引用
    Private Sub OptionButton1_Click(sender As Object, e As EventArgs) Handles OptionButton1.Click
        TextBox2.Visible = False   '隐藏文本框TextBox2
        Label2.Visible = False   '隐藏标签Label2
    End Sub
    0 个引用
    Private Sub OptionButton2_Click(sender As Object, e As EventArgs) Handles OptionButton2.Click
        TextBox2.Visible = False  '隐藏文本框TextBox2
        Label2.Visible = False  '隐藏标签Label2
    End Sub
    0 个引用
    Private Sub OptionButton3_Click(sender As Object, e As EventArgs) Handles OptionButton3.Click
        TextBox2.Visible = True  '显示文本框TextBox2
        Label2.Visible = True    '显示标签Label2
    End Sub
    0 个引用
    Private Sub TextBox3_TextChanged(sender As Object, e As EventArgs) Handles TextBox3.TextChanged
        If Len(TextBox3) > 0 Then '如果录入了数据
            '如果右边一位是"\/?*:<>|"中的任意一位字符
            If Right(TextBox3, 1) = "\" Or Right(TextBox3, 1) = "/" Or Right(TextBox3, 1) = ":" Or Right(TextBox3, 1) = "?" C
                MsgBox "不能使用" & Right(TextBox3, 1) & "对文件命名", vbInformation '提示用户
                TextBox3 = Left(TextBox3, Len(TextBox3) - 1) '删除右边一位字符
            End If
        End If
    End Sub
    0 个引用
    Private Sub CommandButton2_Click(sender As Object, e As EventArgs) Handles CommandButton2.Click
        If OptionButton3 Then  '如果选择了"替换指定字符"
            '如果第二个文本框是空值，那么提示用户，然后结束程序
            If Len(TextBox2) = 0 Then MsgBox "请输入需要替换的名称": Exit Sub
        Else  '否则
            '如果第三个文本框是空值，那么提示用户，然后结束程序
            If Len(TextBox3) = 0 Then MsgBox "请输入替换后的名称": Exit Sub
        End If
        Dim Point_Position As Byte, Bar_Position As Byte, m As Integer, file() As String, FileName As String, ShortName As St
        ReDim file(1 To ListBox1.ListCount)  '根据列表框的元素个数重置变量file的上标
        For m = 0 To ListBox1.ListCount - 1  '遍历列表框的所有元素
            Point_Position = InStr(1, StrReverse(ListBox1.List(m)), ".")   '计算后缀名前面的小圆点的位置
            Bar_Position = InStr(1, StrReverse(ListBox1.List(m)), "\")  '计算文件名称前面的"\"的位置
            If OptionButton1 Then  '如果选择了"插入到原名称之前"
                '生成新的文件名称(在原来的文件名称之前插入TextBox3的值),第一个&之前的字符是文件的路径,TextBox3.Value表示要插
                FileName = Left(ListBox1.List(m), 1 + Len(ListBox1.List(m)) - Bar_Position) & TextBox3.Value & Right(ListBox1
                Name ListBox1.List(m) As FileName  '用Name语句对文件重命名
            ElseIf OptionButton2 Then  '如果选择了"插入到原名称之后"
                '生成新的文件名称(在原来的文件后缀名之前插入TextBox3的值)
                FileName = Left(ListBox1.List(m), Len(ListBox1.List(m)) - Point_Position) & TextBox3.Value & Right(ListBox1.L
                Name ListBox1.List(m) As FileName  '用Name语句对文件重命名
            Else  '否则
                '记录文件的短名称(不包含后缀名)
                ShortName = Left(Dir(ListBox1.List(m)), Len(Dir(ListBox1.List(m))) - Point_Position)
                '生成新的文件名称(在原来的文件后缀名之前插入TextBox3的值)
                FileName = Left(ListBox1.List(m), 1 + Len(ListBox1.List(m)) - Bar_Position) & Replace(ShortName, TextBox2.Val
                Name ListBox1.List(m) As FileName  '用Name语句对文件重命名
            End If
            file(m + 1) = FileName  '将新文件名称导入到数组中
        Next m
        ListBox1.List = WorksheetFunction.Transpose(file)  '更新列表框的内容
    End Sub
    0 个引用
    Private Sub CommandButton3_Click(sender As Object, e As EventArgs) Handles CommandButton3.Click
        Unload Me  '关闭窗体
    End Sub
End Class
```

图 18–7　错误代码提示

7．将 VBA 插件中关于“文件批量命名”过程的代码复制到以上事件过程外壳之中，得到的结果如图 18-8 所示。

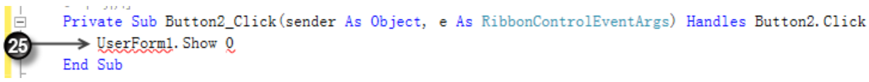


图 18–8　错误代码提示

关于窗体“文件批量命名”的准备工作结束，接下来的工作是修改错误代码。

18.3.2　修改代码

图 18-7 和图 18-8 中罗列了 25 个代码错误提示，现一一展示修改过程。

错误编码：1

修改结果：

```
    Filearr = app.GetOpenFilename("所有文件(*.* ) ,*.*", , , , True)
```

修改依据：VS004

错误编码：2

修改结果：

```
    Dim Arr(Filearr.LongLength - 1) As Object
    Filearr.CopyTo(Arr, 0)
    ListBox1.Items.AddRange(Arr)
```

修改依据：VS043

错误编码：3

修改结果：

```
    TextBox1.Text = Replace(Filearr(1), Dir(Filearr(1)), "")
```

修改依据：VS032

需要注意的是，此处 Filearr 的下标是 1，不是 0

错误编码：4

修改结果：

```
        If Strings.Right(TextBox3.Text, 1) = "\" Or Strings.Right(TextBox3.Text, 1) = "/" Or
Strings.Right(TextBox3.Text, 1) = ":" Or Strings.Right(TextBox3.Text, 1) = "?" Or
    Strings.Right(TextBox3.Text, 1) = "*" Or
    Strings.Right(TextBox3.Text, 1) = "<" Or
    Strings.Right(TextBox3.Text, 1) = ">" Or
    Strings.Right(TextBox3.Text, 1) = "|" Then
```

修改依据：VS013 和 VS005

错误编码：5

修改结果：

```
    MsgBox("不能使用" & Strings.Right(TextBox3.Text, 1) & "对文件命名", vbInformation)
```

修改依据：VS014、VS013 和 VS005

错误编码：6

修改结果：

```
    TextBox3.Text = Strings.Left(TextBox3.Text, Len(TextBox3.Text) - 1)
```

修改依据：VS013 和 VS005

错误编码：7

修改结果：

```
If OptionButton3.Checked Then
```

修改依据：VS005

错误编码：8

修改结果：

```
If Len(TextBox2.Text) = 0 Then MsgBox("请输入需要替换的名称") : Exit Sub
```

修改依据：VS014（此处需要注意一点：尽管 VSTO 没有通过红色波浪线提示代码 Len(TextBox3)有错，但也应该意识到代码的问题所在，它缺少属性，必须补充完整才能运算）。

错误编码：9

修改结果：

```
If Len(TextBox3.Text) = 0 Then MsgBox("请输入替换后的名称") : Exit Sub
```

修改依据：VS014

错误编码：10

修改结果：

```
ReDim file(0 To ListBox1.Items.Count - 1)
```

同时，由于修改了 File 变量的下标，那么对变量赋值的语句也需要做相应的修改。即后面的代码“file(m + 1) = FileName”应该修改为“file(m) = FileName”。

修改依据：VS023 和 VS045

错误编码：11

修改结果：

```
For m = 0 To ListBox1.Items.Count - 1
```

修改依据：VS023 和 VS045

错误编码：12

修改结果：

```
Point_Position = InStr(1, StrReverse(ListBox1.Items(m)), ".")
```

修改依据：VS044

错误编码：13

修改结果：

```
Bar_Position = InStr(1, StrReverse(ListBox1.Items(m)), "\")
```

修改依据：VS044

错误编码：14

修改结果：

```
If OptionButton1.Checked Then
```

修改依据：VS005

错误编码：15

修改结果：

```
FileName = Strings.Left(ListBox1.Items(m), 1 + Len(ListBox1.Items(m)) – Bar_Position) & TextBox3.Text & Strings.Right(ListBox1.Items(m), Bar_Position - 1)
```

修改依据：VS044、VS013 和 VS032

错误编码：16

修改结果：

```
FileSystem.Rename(ListBox1.Items(m), FileName)
```

修改依据：VS044 和 VS057

错误编码：17

修改结果：

```
ElseIf OptionButton2.Checked Then
```

修改依据：VS005

错误编码：18

修改结果：

```
FileName = Strings.Left(ListBox1.Items(m), Len(ListBox1.Items(m)) –Point_Position) & TextBox3.Text & Strings.Right(ListBox1.Items(m), Point_Position)
```

修改依据：VS044、VS013 和 VS032

错误编码：19

修改结果：

```
FileSystem.Rename(ListBox1.Items(m), FileName)
```

修改依据：VS044 和 VS057

错误编码：20

修改结果：

```
ShortName = Strings.Left(Dir(ListBox1.items(m)), Len(Dir(ListBox1.items(m))) - Point_Position)
```

修改依据：VS044 和 VS013

错误编码：21

修改结果：

```
FileName = Strings.Left(ListBox1.Items(m), 1 + Len(ListBox1.Items(m)) - Bar_Position) &
Replace(ShortName, TextBox2.Text, TextBox3.Text) & Strings.Right(ListBox1.Items(m), Point_Position)
```

修改依据：VS044、VS013 和 VS032

错误编码：22

修改结果：

```
FileSystem.Rename(ListBox1.Items(m), FileName)
```

修改依据：VS044 和 VS057

错误编码：23

修改结果：

```
ListBox1.Items.Clear()
ListBox1.Items.AddRange(file)
```

修改依据：VS042

错误编码：24

修改结果：

```
Me.Close()
```

修改依据：VS039

错误编码：25

修改结果：

```
Dim f As New Form1
f.Show()
```

修改依据：VS031

最后需要补充一点，在 TextBox3_Text 事件的第一句代码“If Len(TextBox3) > 0 Then”并没有错误提示，但它其实是错误的，需要将“TextBox3”修改为“TextBox3.Text”。这同时也说明 VSTO 并不一定能把所有错误都标示出来，需要读者熟读书中第 17 章的 69 个差异说明，然后判断代码中是否有遗漏的错误。

当然也可以直接执行代码，当代码出错时，VSTO 会指出错误代码所有行，从而直接找到错

误并加以修改。

在初学时修改代码会比较慢，因为不熟悉这些差异。实际上一周后或者修改过三五次代码后就会发现，69 个问题并非都会出现，而是集中在其中 10 个之间，修改几次后就会对这些差异了然于胸，看到代码直接就修改好了，不需要每次都得翻书。

18.4　升级事件过程“零值控制器”

“零值控制器”和前两个工具稍有不同，它既需要实现零值的控制，又需要实现菜单图标和菜单文字控制，让菜单根据零值的显示与否做出相应变化。

在 VBA 中要通过回调控制菜单的图标和文字变化的过程比较烦琐，而 VSTO 大大简化了回调过程，因此对于功能区的回调代码不需要像前面的操作那样复制 VBA 代码到 VSTO 中，然后再修改代码，而是直接在 VSTO 中书写代码。在本书的 17.6.2 节中有代码模板，参考该代码稍加修改即可。

18.4.1　编写功能区回调过程

“Excel 精灵”插件第三个按钮要实现的功能包含两方面，一是控制零值的显示状态，二是通过菜单的文字和图标来反映零值的状态。对于后者，需要使用以下步骤。

1．双击解决方案管理器中的“Ribbon1.vb”进入功能区的设计窗口，然后双击功能区的第 3 个按钮，从而进入功能区的代码窗口，此时在窗口中会自动生成 Button3_Click 事件的过程外壳。

2．在事件过程的外壳中录入切换按钮状态的代码。

```
    Private Sub Button3_Click(sender As Object, e As RibbonControlEventArgs) Handles Button3.Click
    '引用第三个按钮
    Dim Btn1 As Microsoft.Office.Tools.Ribbon.RibbonButton = Globals.Ribbons.Ribbon1.Tab1.Groups(0).Items(2)
    '使用 Not 来控制零值，表示如果已显示就切换为不显示，如果不显示则切换为显示。
    app.ActiveWindow.DisplayZeros = Not app.ActiveWindow.DisplayZeros
    '如果已经显示零值则菜单显示“已显示零值”,否则显示“未显示零值”
    Btn1.Label = IIf(app.ActiveWindow.DisplayZeros, "已显示零值", "未显示零值")
    '如果已经显示零值则菜单图标采用 AcceptInvitation,否则用 DeclineInvitation
        Btn1.OfficeImageId = IIf(app.ActiveWindow.DisplayZeros,
     "AcceptInvitation", "DeclineInvitation")
End Sub
```

18.4.2　编写应用程序级事件过程

VBA 插件“Excel 精灵”用到了 Workbook_Open、app_SheetActivate、app_WorkbookActivate、app_WorkbookOpen 等四个应用程序级事件，在 VSTO 中也需要用这四个事件，只是事件的名称和参数稍有不同。具体的操作步骤如下。

1．在“解决方案资源管理器”中双击“ThisAddIn.vb”，从而进入代码窗口。

2．在窗口中默认已经存在 ThisAddIn_Startup 和 ThisAddIn_Shutdown 两个事件，其中 ThisAddIn_Startup 事件对应前面的 Workbook_Open 事件，因此只需要添加其余三个事件即可。添加 Excel 的应用程序事件的办法是：在代码窗口上方的导航栏中单击第二个下拉列表，从中选择对象 Application，然后进入第三个下拉列表中分别选择 SheetActivate、WorkbookActivate 和 WorkbookOpen，从而添加三个事件过程外壳，组合成完整的事件过程。

3．对 ThisAddIn_Startup、Application_SheetActivate、Application_WorkbookActivate 和 Application_WorkbookOpen 四个事件添加同样的三句代码，添加后将得到如下结果：

```
    Private Sub ThisAddIn_Startup() Handles Me.Startup          '加载插件时触发
        Dim Button_1 As Microsoft.Office.Tools.Ribbon.RibbonButton =
Globals.Ribbons.Ribbon1.Tab1.Groups(0).Items(2) '引用第三个按钮
        '如果窗口中未显示零值，那么按钮显示为“不显示零值”，否则显示为“已显示零值”
        Button_1.Label = IIf(app.ActiveWindow.DisplayZeros = False, "不显示零值", "已显示零值")
        '如果窗口中已显示零值，那么按钮的图标采用“AcceptInvitation”，否则图标采用“DeclineInvitation”
        Button_1.OfficeImageId = IIf(app.ActiveWindow.DisplayZeros, "AcceptInvitation", "DeclineInvitation")
    End Sub
    '激活工作表时触发
    Private Sub Application_SheetActivate(Sh As Object) Handles Application.SheetActivate
        Dim Button_1 As Microsoft.Office.Tools.Ribbon.RibbonButton =
Globals.Ribbons.Ribbon1.Tab1.Groups(0).Items(2)
        Button_1.Label = IIf(app.ActiveWindow.DisplayZeros = False, "不显示零值", "已显示零值")
        Button_1.OfficeImageId = IIf(app.ActiveWindow.DisplayZeros, "AcceptInvitation", "DeclineInvitation")
    End Sub
    Private Sub Application_WorkbookActivate(Wb As Workbook) Handles Application.WorkbookActivate
    '激活工作簿时触发
        Dim Button_1 As Microsoft.Office.Tools.Ribbon.RibbonButton =
Globals.Ribbons.Ribbon1.Tab1.Groups(0).Items(2)
        Button_1.Label = IIf(app.ActiveWindow.DisplayZeros = False, "不显示零值", "已显示零值")
        Button_1.OfficeImageId = IIf(app.ActiveWindow.DisplayZeros, "AcceptInvitation", "DeclineInvitation")
    End Sub
    '打开工作簿时触发
    Private Sub Application_WorkbookOpen(Wb As Workbook) Handles Application.WorkbookOpen
        Dim Button_1 As Microsoft.Office.Tools.Ribbon.RibbonButton =
Globals.Ribbons.Ribbon1.Tab1.Groups(0).Items(2)
        Button_1.Label = IIf(app.ActiveWindow.DisplayZeros = False, "不显示零值", "已显示零值")
        Button_1.OfficeImageId = IIf(app.ActiveWindow.DisplayZeros, "AcceptInvitation", "DeclineInvitation")
    End Sub
```

三句代码的含义分别是声明一个代表功能区按钮的变量，赋值为 Ribbon1 的第 3 个按钮；然后根据零值的显示状态去设置该按钮要显示的文本；最后根据零值的显示状态去设置该按钮的图标名称。

经过以上设置后，插件已经开发完成，最后一步则是生成 DLL 文件。

4．选择菜单“生成”→“生成 Excel 精灵”，然后关闭 Visual Studio 2015。

此时进入“C:\Excel 精灵\Excel 精灵\bin\Debug”路径可以看到刚生成的所有文件。

本例案例文件请参考：..\ 第 18 章\Excel 精灵\

18.5 打包安装程序

如果打包安装程序的所有过程都手动操作一次，那么可能 10 分钟也无法完成，但是本书的教学是以模板为基础的，只要按模板操作，5 秒钟以内即可打包完成。

18.5.1 制作安装程序

将 VSTO 插件打包需要一个必要文件“安装文件模板.iss”和两个非必要文件“1.ico”和

“WizModernImage.bmp”。在本书第 16 章中提供了这 3 个文件，具体操作办法如下。

1．将“安装文件模板.iss”、“1.ico”和“WizModernImage.bmp”三个文件复制到“C:\Excel 精灵\Excel 精灵\bin\Debug”路径下。

其中后两个非必要文件是否需要复制由读者自行决定，这两个文件的功能仅仅是让安装程序稍微美观一点或者加以区分而已，并没有实际的功能。在复制前也可以对它们稍加修改，更换一下图片内容。

2．双击打开“安装文件模板.iss”，然后按【Ctrl+h】组合键打开“替换”对话框，然后将“玩转 VSTO”全部替换成“Excel 精灵”，设置界面如图 18-9 所示。

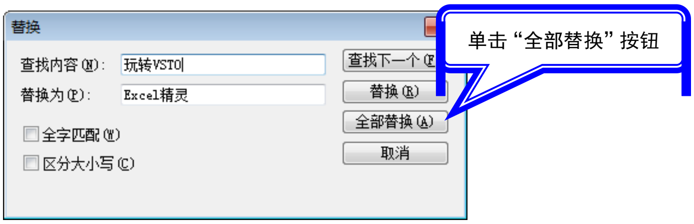


图 18–9　替换插件名称

3．选择 Inno Setup 的菜单“构建”→“编译”，从而生成名为“Setup.exe”的安装程序。

4．将“Setup.exe”和“运行环境.exe”压缩成一个“Excel 精灵.rar”。如果用户的 Windows 版本低于 Windows 8 或者 Office 版本低于 2013，那么需要安装“运行环境.exe”。

18.5.2　安装测试

假设本机是 Windows 10 以及 Office 2016，安装过程如下。

1．双击“Excel 精灵.rar”中的“Setup.exe”文件，将自动弹出如图 18-10 所示的提示页面，由于当前系统不需要安装运行环境，因此直接单击“确定”按钮即可，程序会进入安装向导的第一页，界面如图 18-11 所示。

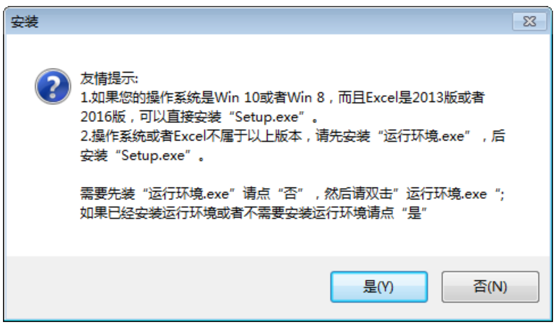


图 18–10　提示信息

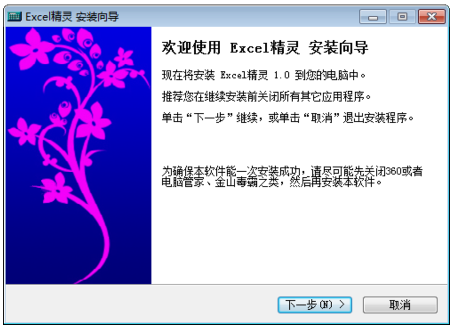


图 18–11　安装向导第一页

2．单击“下一步”按钮，进入安装向导第二页，此页用于设置安装路径，一般保持默认路径即可，直接单击“下一步”按钮，进入“准备安装”页面，如图 18-12 所示。

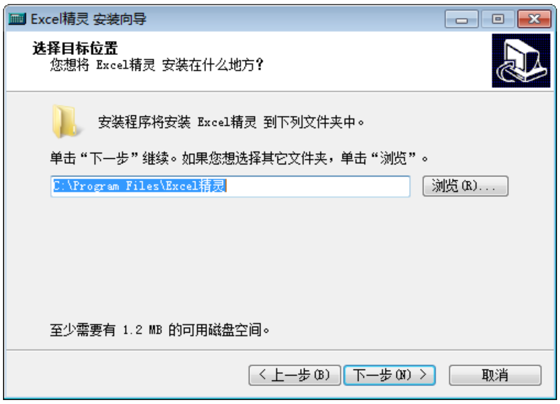


图 18–12　选择安装路径

3．单击“安装”按钮，然后安装程序会进入最后的“安装向导完成”页面，单击“完成”按钮关闭对话框，从而完成安装。

4．打开 Excel 查看插件是否安装成功。首次打开时一般会看到图 18-13 所示的对话框，以后再打开 Excel 时则不再出现该对话框。此时必须单击其中的“安装”按钮，否则插件不会注册成功，如图 18-13 所示。

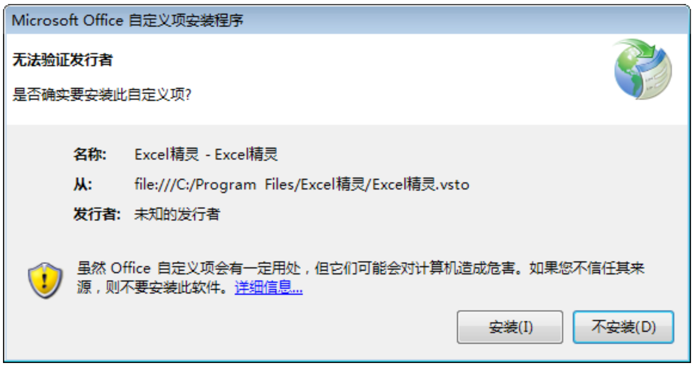


图 18–13　提示是否安装插件

5．如果用户使用的是 WPS，那么打开 WPS 后可以看到如图 18-14 所示的界面；如果用户使用的是 Excel 2016，那么打开 Excel 2016 后将看到如图 18-15 所示的界面，表明安装成功。

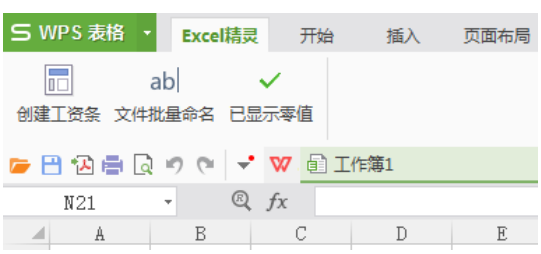


图 18–14　插件在 WPS 中的界面

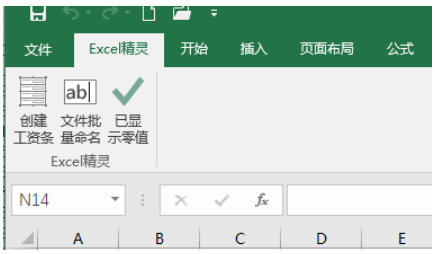


图 18–15　插件在 Excel 中的界面

本例案例文件请参考：..\ 第 18 章\Excel 精灵\ Excel 精灵\bin\Debug\ Excel 精灵.rar

18.6　课后思考

1．要如何修改以下 VBA 代码，才能使其在 VSTO 中正常执行?

```
Range("B11").EntireRow.Insert , CopyOrigin:=xlFormatFromLeftOrAbove
```

2．在对功能区按钮添加图标时，如果对 OfficeImageId 属性赋值可以让按钮引用 Excel 的内置图标，那么如何才能让功能区按钮显示为自定义的图标? 例如 C 盘中的 ABC.jpg。

3．ThisAddIn_Startup 事件和 Application_WorkbookOpen 事件的区别是什么?

4．Ribbon1 的代码窗口中的调用功能区按钮和模块中的调用功能区按钮有何区别?

5．如何才能让 Excel 插件可以在 WPS 中使用?

第 19 章　VSTO 的更多高级应用

VBA 和 VSTO 都是在 VB 的基础上发展而来的，并加入了一些新特性。其中 VSTO 在高速发展，几乎每年都会加入很多全新的功能。本书既然已经提到借助 VSTO 来封装 VBA 代码，那么顺便介绍几个 VSTO 比较优秀并且 VBA 较为欠缺的功能吧。

19.1　添加窗体状态栏

StatusStrip 控件属于 ControlTipText 的升级版（ControlTip Text 是 VBA 控件的一个属性，用于生成屏幕提示信息），都可用于为其他控件添加说明，从而提升窗体的直观性和实用性。不过 StatusStrip 控件的一些新特性注定了它将会完全取代 ControlTipText，甚至也会取代 ToolTip 控件（属于 VB.net 的专用控件）。

19.1.1　设计

StatusStrip 控件相对于 ControlTipText 控件有诸多优势，一是鼠标移过控件时不需要等待就会产生提示信息，二是提示信息不会覆盖控件，三是可以在窗体的状态栏中显示其他的信息，包含软件公司名称、地址或者当前时间等，从而使窗体看起来有更加专业。

在窗体中添加 StatusStrip 控件的步骤如下。

1．打开 Visual Studio 2015 软件，新建一个“Excel 2013 和 2016 VSTO 外接程序”，将上方的.Net Framework 版本号由 4.5.2 修改为 4，并且取名为“窗体任务栏模板”。

2．选择菜单“项目”→“添加 Windows 窗体”，然后在弹出的对话框中单击“添加”按钮。

3．在窗体中添加 1 个列表框和 3 个命令按钮，并按图 19-1 所示的方式排列。

4．将工具箱中的 StatusStrip 控件拖到窗体中，在窗体底部会出现一个状态栏，效果如图 19-2 所示。

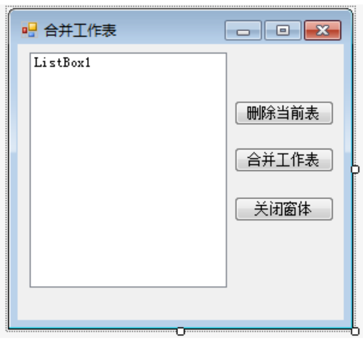


图 19－1　列表框与命令按钮布局

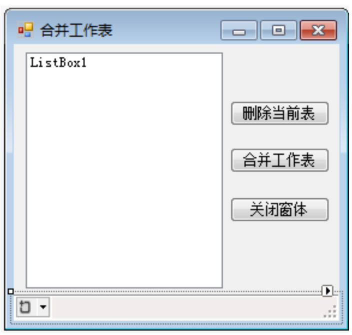


图 19－2　StatusStrip 控件

5．双击窗体或者任意控件进入窗体的代码窗口，然后删除自动生成的事件，重新录入以下

新的事件过程：

```
'加载窗体时触发事件
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    For Each sht As Excel.Worksheet In app.Sheets
         ListBox1.Items.Add(sht.Name)
     Next sht
End Sub
Private Sub Form1_MouseMove(sender As Object, e As MouseEventArgs) Handles
Me.MouseMove                    '鼠标指针移过窗体时触发事件
  Me.StatusStrip1.Items.Clear()    '清除以前的信息
  Me.StatusStrip1.Items.Add("四维软件公司    " & DateAndTime.Now)       '添加新的信息
End Sub
Private Sub Button1_MouseMove(sender As Object, e As MouseEventArgs) Handles
Button1.MouseMove               '鼠标指针移过按钮 1 时触发事件
  Me.StatusStrip1.Items.Clear()    '清除以前的信息
  Me.StatusStrip1.Items.Add("单击此按钮删除列表框中选中的工作表")       '添加新的信息
End Sub
Private Sub Button2_MouseMove(sender As Object, e As MouseEventArgs) Handles
Button2.MouseMove               '鼠标指针移过按钮 2 时触发事件
  Me.StatusStrip1.Items.Clear()    '清除以前的信息
  Me.StatusStrip1.Items.Add("单击此按钮合并列表中的所有工作表")       '添加新的信息
End Sub
Private Sub Button3_MouseMove(sender As Object, e As MouseEventArgs) Handles
Button3.MouseMove               '鼠标指针移过按钮 3 时触发事件
  Me.StatusStrip1.Items.Clear()    '清除以前的信息
  Me.StatusStrip1.Items.Add("单击此按钮关闭窗体")       '添加新的信息
End Sub
Private Sub ListBox1_MouseMove(sender As Object, e As MouseEventArgs) Handles
ListBox1.MouseMove              '鼠标指针移过列表框时触发事件
  Me.StatusStrip1.Items.Clear()    '清除以前的信息
  Me.StatusStrip1.Items.Add("列表中包含所有工作表,可以删除不需要合并的表")'添加新的信息
End Sub
```

其中“Form1_Load”事件与 StatusStrip 事件无关，该事件用于添加工作表名到列表框中，其他 5 个事件则与 StatusStrip 相关，分别表示当鼠标移过窗体、列表框以及 3 个命令按钮时添加提示信息到窗体的状态栏。

在写以上几个事件过程时一定要注意，事件过程的外壳不能手写，而是在导航栏中选择控件的名称和事件名称，从而让 VB.net 自动生成完整的事件过程外壳，然后书写事件过程的代码。

窗体中的 3 个命令按钮的代码与本章的主题无关，因而省略。设计功能区菜单以及在模块中声明变量的过程和本书第 16 章、第 18 章的操作完全一致，因此也省略。

本例案例文件请参考：..\ 第 19 章\窗体任务栏模板\

19.1.2　测试

测试外接程序有两种方法，一是直接按【F5】键启动 Excel 2016，然后单击菜单测试代码；二是生成 DLL 插件，并按第 18 章的方法生成 exe 安装文件，然后安装好插件后再测试。

本节按方法一操作，步骤如下。

1．在 Visual Studio 2015 中按下【F5】键，从而打开 Excel 2016（如果在新建项目时选择了“Excel 2010 VSTO 外接程序”，那么将会打开 Excel 2010）。

2．在 Excel 2016 界面中有一个名为“合并工作表”的菜单，如图 19-3 所示，单击该菜单可以打开“合并工作表”窗体。

3．将鼠标移向按钮“合并工作表”，在窗体的状态栏会看到如图 19-4 所示的提示文字。

4．将鼠标移向窗体空白区域，在窗体的状态栏会看到如图 19-5 所示的提示文字。

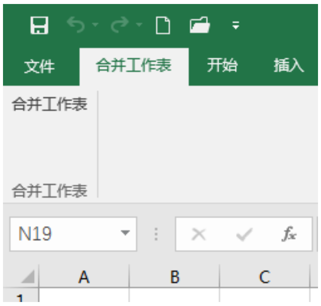


图 19–3　功能区菜单

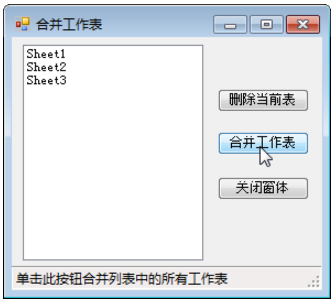


图 19–4　按钮提示

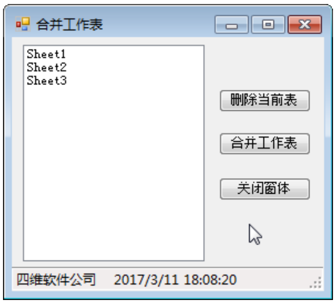


图 19–5　窗体提示

关于 StatusStrip 控件需要补充以下三点。

其一，为 StatusStrip 添加说明文字之前需要清除以前的文字，因为新的文字不会覆盖以前的文字，而是追加在后方。

其二，StatusStrip.Items.Add 的语法如下：

```
StatusStrip.Items.Add(String, Image)
```

第一参数表示要显示的文本，第二参数表示要显示的图标，两者既可以同时使用，也以只使用其中一个参数。当需要在状态栏显示图标时，通常的做法是在窗体中添加一个 PictureBox 控件，并为该控件指定一幅图片，然后用“PictureBox1.Image”作为 StatusStrip1.Items.Add 的参数即可。

其三，当状态栏显示的文本超过窗体的宽度时会显示为空白，因此需要控制文本长度。

19.2　创建任务栏图标

多数软件在运行时都会在桌面右下角处显示一个代表软件状态的图标，例如 QQ、邮箱等软件。在使用 VSTO 设计 Excel 插件时可以通过 NotifyIcon 控件为窗体或者功能区添加任务栏图标，该图标既可用于说明软件的状态，也可以弹出文字，提示软件的执行进度或者警告信息。

19.2.1　设计

NotifyIcon 控件既能用于窗体也能用于功能区，本节以功能区为例演示设计过程，步骤如下。

1．打开 Visual Studio 2015 软件，新建一个“Excel 2013 和 2016 VSTO 外接程序”，将上方的.Net Framework 版本号由 4.5.2 修改为 4，并且取名为“任务栏图标”。

2．选择菜单“项目”→“添加组件”，选择“功能区(可视化设计)”，然后单击“添加”按钮。

3．在生成的功能区的右方或者下方单击右键，选择快捷菜单中的“删除命令”，从而删除内置的选项卡。

4．进入工具箱，分别将 Tab 控件、Group 控件和 Button 控件拖到功能区中，按图 19-6 所示的方式排列，并且设置 Label 属性。将其中两个按钮的 OfficeImageId 属性分别设置为“AcceptInvitation”和“AddressBook”，并且将 ControlSize 属性设置为“RibbonControlSizeLarge”。

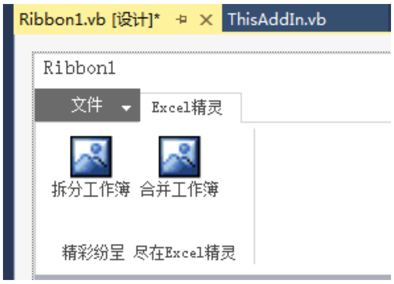


图 19–6　功能区控件布局

5．进入工具箱，单击“所有 Windows 窗体”，找到 NotifyIcon 控件，将它拖曳到功能区中。NotifyIcon 控件并不是显示在功能区中的，而是在下方靠近“错误列表”处显示“NotifyIcon1”。

6．双击功能区中的按钮 1，从而进入代码窗口，然后录入以下三个事件过程代码（事件过程的外壳不要手动录入，而是从导航栏中选择对象名称和事件名称，从而自动生成）。

```
Private Sub Button1_Click(sender As Object, e As RibbonControlEventArgs)
Handles Button1.Click
    NotifyIcon1.Visible = True                                        '显示任务栏图标
    NotifyIcon1.BalloonTipIcon = System.Windows.Forms.ToolTipIcon.Error  '在气球上显示错误图标
    NotifyIcon1.Icon = Drawing.SystemIcons.Question                   '在任务栏显示系统问号图标
    NotifyIcon1.BalloonTipTitle = "拆分工作簿"                          '指定提示信息的标题
    NotifyIcon1.BalloonTipText = "拆分期间，请不要关闭哦，拜托!"             '指定提示信息的正文
    NotifyIcon1.Text = "拆分工作簿"                                     '指定任务栏图标的名称
    NotifyIcon1.ShowBalloonTip(30000)    '指定提示信息的显示时间，单位为毫秒
    ' 更多代码……
End Sub
Private Sub Button2_Click(sender As Object, e As RibbonControlEventArgs) Handles Button2.Click
    NotifyIcon1.Visible = True                                        '显示任务栏图标
    NotifyIcon1.BalloonTipIcon = System.Windows.Forms.ToolTipIcon.Info   '在气球上显示信息图标
    NotifyIcon1.Icon = Drawing.SystemIcons.Shield                     '在任务栏显示盾牌图标
    NotifyIcon1.BalloonTipTitle = "合并工作簿"                          '指定提示信息的标题
    NotifyIcon1.BalloonTipText = "合并期间，请不要关闭哦，拜托!"             '指定提示信息的正文
    NotifyIcon1.Text = "合并工作簿"                                     '指定任务栏图标的名称
    NotifyIcon1.ShowBalloonTip(30000)    '指定提示信息的显示时间，单位为毫秒
    ' 更多代码……
End Sub
'关闭任务栏提示信息时，隐藏图标
Private Sub NotifyIcon1_BalloonTipClosed(sender As Object, e As EventArgs)
Handles NotifyIcon1.BalloonTipClosed
    NotifyIcon1.Visible = False
End Sub
```

其中第三个过程“NotifyIcon1_BalloonTipClosed”不是必要的过程，它的功能是在任务栏的文字信息关闭时自动隐藏图标，如果删除此事件过程则不会隐藏图标。

Drawing.SystemIcons.Question 表示系统内置的问号图标，Drawing.SystemIcons.Error 表示系统内置的错误图标，Drawing.SystemIcons.Shield 表示系统内置的盾牌图标，Drawing.SystemIcons.Warning 表示系统内置的警告图标，Drawing.SystemIcons.Information 则表示系统内置的信息图标。当然，也可以使用自定义的图标。

要注意 System.Windows.Forms.ToolTipIcon.Error 用于指定气泡中的图标，而 Drawing.SystemIcons.Error 则用于指定任务栏图标，两者的位置不同。

本例案例文件请参考：..\ 第 19 章\任务栏图标\

19.2.2 测试

测试外接程序有两种方法，一是直接按【F5】键启动 Excel 2016，然后单击菜单测试代码，二是生成 DLL 插件，并按第 18 章的方法生成 exe 安装文件，然后安装好插件后再测试。

本节按第一种方法操作，步骤如下。

1．在 Visual Studio 2015 中按下【F5】键，从而打开 Excel 2016（如果在新建项目时选的是“Excel 2010 VSTO 外接程序”，那么将会打开 Excel 2010）。

2．在 Excel 2016 界面中有一个名为“Excel 精灵”的选项卡，包含两个按钮，如图 19-7 所示。单击菜单“拆分工作簿”，在 Windows 的任务栏中将产生如图 19-8 所示的气泡图，3 秒钟后会自动关闭。

3．单击菜单“合并工作簿”，在 Windows 的任务栏中将产生如图 19-9 所示的气泡图。

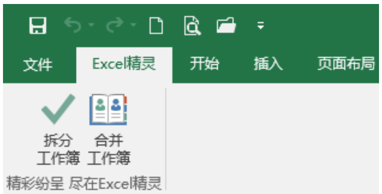


图 19–7　功能区按钮布局



图 19–8　拆分工作簿时的任务栏图标

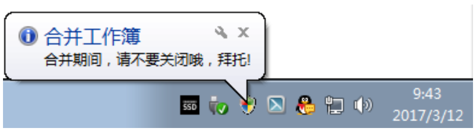


图 19–9　合并工作簿时的任务栏图标

本例主要用于测试任务栏图标，因此省略了拆分工作簿和合并工作簿的相关代码。

19.3 自动发邮件

通过 Excel VBA 引用 Outlook 可以向外发送邮件，也可以引用 CDO.Message 来发邮件，但这些都属于调用外置功能，而且代码相当复杂。使用 VSTO 开发 Excel 插件时可以调用内置的 MailMessage 对象来发送邮件，代码简单且易于理解。

19.3.1 设计

发送邮件需要使用发件人邮箱地址、密码、SMTP 服务器地址、邮件标题、邮件正文和接收人邮箱地址等信息，因此通常需要配合表格或者窗体使用。本节以窗体为例演示自动发送邮件的设计过程，步骤如下。

1．打开 Visual Studio 2015 软件，新建一个“Excel 2013 和 2016 VSTO 外接程序”，将上方的.Net Framework 版本号由 4.5.2 修改为 4，并且取名为“任务栏图标”。

2．选择菜单“项目”→“添加 Windows 窗体”，然后在弹出的对话框中单击“添加”按钮。

3．在窗体中添加两个 GroupBox 控件、7 个 Label 控件、6 个 TextBox 控件和一个 Button 控件，并按如图 19-10 所示的方式排列。



图 19-10　窗体控件布局

文本框默认只显示一行，在需要显示多行时进入“属性”对话框中将 Multiline 属性的值由 False 修改为 True。

4．选择存放邮箱密码的文本框 TextBox5，然后在右下角的“属性”对话框中将 PasswordChar 属性设置为“*”，表示用户在录入密码时不显示字符本身，而是显示为对应数量的“*”。

5．双击“发送”按钮，进入窗体的代码窗口，然后录入以下代码：

```
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
    Button1.Click
      Dim message As New System.Net.Mail.MailMessage(TextBox4.Text, TextBox1.Text, TextBox2.Text, TextBox3.
Text)
        Dim emailClient As New System.Net.Mail.SmtpClient(TextBox6.Text)
        emailClient.Credentials = New System.Net.NetworkCredential(TextBox4.Text, TextBox5.Text)
        emailClient.Send(message)          '发送邮箱
        Me.Close()                         '关闭窗体
    End Sub
```

System.Net.Mail.MailMessage 包含 4 个参数，分别代表发件人邮箱、收件人邮件、标题和正文。

System.Net.Mail.SmtpClient 则只有 1 个参数，即 SMTP 服务器地址。

System.Net.NetworkCredential 有 2 个参数，分别是发件人的邮箱地址和密码。

6．在解决方案工资源管理器中双击“ThisAddIn.vb”，删除里面自带的 2 个事件过程外壳，然后录入以下代码：

```
    WithEvents 子菜单 As Office.CommandBarButton '声明带事件的 CommandBarButton 型变量
      Private Sub ThisAddIn_Startup() Handles Me.Startup
      子菜单 = Globals.ThisAddIn.Application.CommandBars(1). Controls.
Add(Microsoft.Office.Core.MsoControlType.msoControlButton, , , , True)   '添加一个工作表菜单
      With 子菜单 '引用新建的这个菜单
            '让菜单同时显示文本与图标
          .Style = Microsoft.Office.Core.MsoButtonStyle.msoButtonIconAndCaption
          .Caption = "发邮件" '指定菜单文字
          .FaceId = 483 '指定菜单的图标
      End With
```

```
End Sub
Private Sub 子菜单_Click(Ctrl As Microsoft.Office.Core.CommandBarButton,
ByRef CancelDefault As Boolean) Handles 子菜单.Click
    Dim f As New Form1
    f.Show() '调用发邮件的窗体
End Sub
```

“子菜单”是公共变量，需要放在过程之外。此外必须用 WithEvents 声明变量，否则它不具备事件功能。

ThisAddIn_Startup 事件会在加载插件时触发，在本例中此事件过程的代码创建了一个传统的菜单命令，将会显示在 Excel 2016 的“加载项”选项卡中。

子菜单_Click 事件则是在单击菜单命令时要触发的事件，在本例中单击菜单时会调用名为 Form1 的窗体。

本例案例文件请参考：..\ 第 19 章\发邮件模板\

19.3.2 测试

测试代码的步骤如下。

1．在 Visual Studio 2015 中按下【F5】键，从而打开 Excel 2016（如果在新建项目时选的是“Excel 2010 VSTO 外接程序”，那么将会打开 Excel 2010）。

2．选择菜单“加载项”→“发邮件”，从而打开“发送邮件”窗口。

3．在窗体中录入收件人地址、邮件标题、邮件正文、发件地址、发件人密码和 SMTP 服务器地址，然后单击“发送”按钮即可开始发送邮件，如果两个邮箱地址正确，并且密码正确、SMTP 服务器地址正确，那么瞬间就会发送成功。

在测试时，笔者的设置信息如图 19-11 所示，而发送成功后，收件人会收到如图 19-12 所示的邮件。



图 19-11 设置发送信息



图 19-12 收到的邮件

19.4 全自动合并数据

VBA 在合并报表、汇总报表方面已经相当强大了，而 VSTO 新增的 FileSystemWatcher 控件可使插件更加强大。本节展示 FileSystemWatcher 控件的实用功能。

19.4.1　设计

公司需求：公司有 7 个车间，每天 7 个车间的文员会将本车间的生产报表“一车间.xlsx”、“二车间.xlsx”、“三车间.xlsx”……发送到总经理助理处，然后总经理助理将 7 个报表中最新一天的数据合并到一起，便于查看、汇总、分析。

解决思路：由于都处于同一个公司的局域网，可以将总经理助理的“E：\合并工作簿”文件夹共享，然后在其他 7 台机器中打开这个共享的文件夹，每天做完生产报表后，将报表复制到这个“合并工作簿中”，总经理助理的电脑中通过插件自动合并存放到此路径下的工作簿的最后一个工作表数据。

由于生产报表是将每天的日报表存放在一个工作表中，那么一个工作簿中会有多个工作表，按日期顺序排列。在合并时只需要合并工作簿的最后一个工作表即可，每天自动合并一次，生成的合并报表也同样将合并结果存在对应的工作表中，按日期顺序排列。

重点和难点在于 7 个车间的文员并不是同时把文件放入“E：\合并工作簿”，而且既有可能上午完成报表，也可能下午完成报表，因此插件需要随时监控文件夹中的文件变化，有几个文件就合并几个，当中途有新文件放入“E：\合并工作簿”时，插件会主动侦测到，然后自动打开该文件、合并并且关闭文件，直到 7 个文件都合并完成后，自动关闭存放结果的工作簿。

简而言之，不管 7 个车间的文员什么时候将文件提交过来，总经理助理只需要打开“产量总表.xlsx”即可，其他一切事项都由插件全自动完成。

图 19-13 和图 19-14 分别是设置与打开共享文件夹的界面。

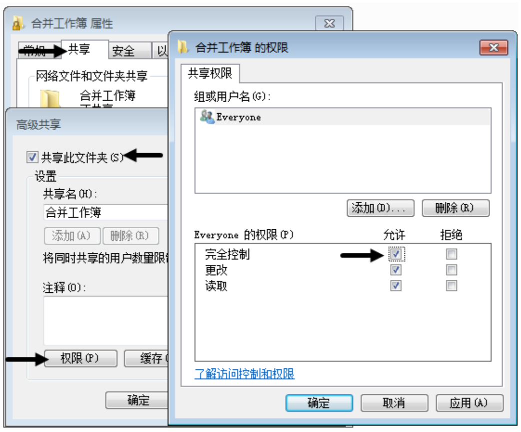


图 19-13　设置共享文件夹

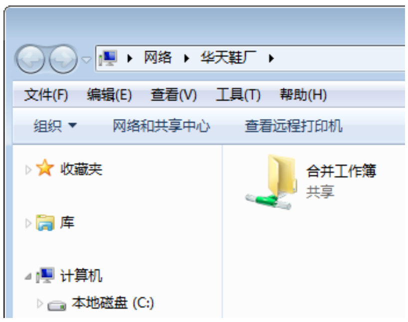


图 19-14　打开共享文件夹

设计插件的过程如下。

1．打开 Visual Studio 2015 软件，新建一个“Excel 2013 和 2016 VSTO 外接程序”，将上方的.Net Framework 版本号由 4.5.2 修改为 4，并且取名为“自动合并报表”。

2．选择菜单“项目”→“添加模块”，然后在弹出的对话框中单击“添加”按钮。

在“Module1.vb”的代码窗口中录入以下代码，用于声明一个公共变量：

```
Public app As Excel.Application = Globals.ThisAddIn.Application
```

3．选择菜单“项目”→“添加 Windows 窗体”，然后在弹出的对话框中单击“添加”按钮。

4．进入工具箱中，将 FileSystemWatcher 控件拖到窗体中（在窗体中看不到控件，只会在下方靠近“错误列表”处显示“FileSystemWatcher1”）。

5．双击窗体进入代码窗口，然后录入以下两个过程和声明公共变量的代码，其中 Form1_Load

是窗体事件过程，应该通过从导航栏中选择对象名称和过程名称的方式生成过程外壳，不宜手动录入。

```
    Dim FileName As String, Item As Byte = 0       '声明几个公共变量
    Dim PathStr As String = "E:\合并工作簿\"        '请根据自己的实际情况修改路径
    Dim Wbk As Excel.Workbook, acWbk As Excel.Workbook = app.ActiveWorkbook
    Dim acSht As Excel.Worksheet, endrow As Int32, newSht As Excel.Worksheet
    '加载窗体时触发此事件
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Me.ShowInTaskbar = False                   '不在任务栏中显示窗体的图标
        FileSystemWatcher1.Path = PathStr          '让 FileSystemWatcher 控件监控"C:\合并工作簿\"路径
        FileSystemWatcher1.Filter = "?车间.xls*"   '监控包含"车间"且名字为 3 个字的 Excel 产量报表
        FileSystemWatcher1.EnableRaisingEvents = True '启用监控
        Me.Top = -10000                            '隐藏窗体，当上边距为-10000 时，将无法看到窗体
        On Error Resume Next                       '程序出错时继续执行下一句代码
        app.DisplayAlerts = True                   '关闭提示
       '如果存在今日日期命名的工作表，那么删除它
        app.Sheets(Format(DateAndTime.Today, "yyyyMMdd")).delete
       '将活动工作簿的最后一个工作表赋值给变量 acSht
        acSht = app.Sheets.Add(, app.Sheets(app.Sheets.Count) )
        acSht.Name = Format(DateAndTime.Today, "yyyyMMdd")  '将最后一个工作表命名为今日日期
        FileName = Dir(PathStr & "?车间.xls*")              '搜索包含“?车间.xls*”的工作簿
        If Len(FileName) = 0 Then Exit Sub                  '如果找不到就结束过程
        app.ScreenUpdating = False                          '关闭屏幕更新，从而提升代码执行效率
        Do '开始循环
            Wbk = app.Workbooks.Open(PathStr & FileName, False) '打开找到的文件
             newSht = Wbk.Sheets(Wbk.Sheets.Count)          '将最后一个工作表赋值给变量 newSht
            '如果最后一个工作表的已用区域大于 2 行（表示不是空表）
             If newSht.UsedRange.Rows.Count >= 2 Then
                 Item = Item + 1                            '累加计数器，从而了解合并了几个工作簿
                 合并(PathStr & FileName, acSht, newSht)     '调用过程“合并”，包含三个参数
                 app.CutCopyMode = False                    '取消剪切模式
             End If
             Wbk.Close(False)               '关闭工作簿（已经合并完成）
             FileName = Dir()               '搜索下一个符合条件的工作簿
    Loop Until Len(FileName) = 0            '一直循环，直到找不到新的符合条件的目标时为止
    app.Cells.EntireColumn.AutoFit()        '自动调整列宽
    app.ScreenUpdating = True               '恢复屏幕更新
    If Item = 7 Then                        '已经合并了 7 个报表
            '删除“E:\合并工作簿”所有文件
             Shell("cmd.exe /c del e:\合并工作簿\*.*   /q", AppWinStyle.MinimizedNoFocus)
             Me.Close() '那么关闭窗体
            '将最后一个工作表加上“合并完成”
             acSht.Name = Format(DateAndTime.Today, "yyyyMMdd 已合并完成")
             app.ActiveWorkbook.Save()      '保存工作簿
             app.ActiveWorkbook.Close()     '关闭工作簿
        End If
    End Sub
    '过程“合并”带有 3 个参数，第一参数代表要合并的工作簿名称，包含路径；第二参数代表存放结果的工作表，
'第三参数代表被合并的工作表
    Sub 合并（Filename As String, sht1 As Excel.Worksheet, sht2 As Excel.Worksheet）
```

```
            '记录要合并的工作表的最后一个非空行的行号
             endrow = sht2.Cells.Find("*", sht2.Cells(1, 1), -4163, 1, 1, 2).Row
             If Item = 1 Then '如果合并第一个工作表
                 '将 sht2 中有数据的区域复制到 Sht1 的 B1（A 列用于存放工作簿名称）
                 app.Intersect(sht2.Rows("1:" & endrow), sht2.UsedRange).Copy(sht1.Cells(2))
                 sht1.Range("a1").Value = "来自工作簿" '对 A1 赋值标题
                '引用 sht1 的 A2 到 A 列 endrow - 1 行，其中 endrow 等于 Sht2 的数据行数
                 With sht1.Cells(2, 1).resize(endrow - 1, 1)
                     .Merge()    '将该区域合并
                     .value = Wbk.Name '向该区域写入被合并的工作簿名称
                 End With
            Else '如果不是合并第一个表
                '引用 sht1A 列中第一个非空单元格向下的数量等于 endrow - 1 的区域
                '其中 endrow  等于 Sht2 的数据行数
                 With sht1.Cells(sht1.Rows.Count, 2).end(-4162).offset(1, -1).resize(endrow - 1, 1)
                     .Merge '将该区域合并
                     .value = Wbk.Name '向该区域写入被合并的工作簿名称
                 End With
           '将 sht 中除掉第一行标题行后的其他有数据的区域复制到 sht1 的 B 列最后一个非空行下方
                 app.Intersect(sht2.Rows("2:"  &  endrow),  sht2.UsedRange ) .Copy(sht1.Cells(sht1.Rows.Count,
2).end(-4162).offset(1, 0))
            End If
        End Sub
```

其中 Form1_Load 事件的作用是加载窗体时通过 Dir 与 Do…Loop 组合在“E:\合并工作簿”中搜索“?车间.xls*”，如果找到有符合条件的工作簿就打开它，并且调用过程“合并”复制该工作簿的最新一个工作表中的数据，存放到“产量总表.xlsx”中新建的工作表中。在合并的同时计算合并数量，如果数量等于 7 则表示合并完成，此时删除 7 个工作簿，然后保存并关闭“产量总表.xlsx”，并且在“产量总表.xlsx”中标注合并完成，避免当天再次打开工作簿时再次执行合并。

如果在打开“产量总表.xlsx”时，“E：\合并工作簿”中的文件不足 7 个，那么通过 FileSystemWatcher1 控件监测“E：\合并工作簿”中的文件变化，当有新文件复制进来时，插件需要自动合并新的文件，此时以上过程就无法完成了，因此需要配合以下代码使用（参见第 6 步和第 7 步）。

6．在窗体代码窗口导航栏中间的列表中选择对象 FileSystemWatcher1，再在事件列表中选择事件过程 Created，从而在窗口中自动生成 FileSystemWatcher1_Created 事件的过程外壳。

7．将 FileSystemWatcher1_Created 事件代码补充完整。

```
        Private  Sub  FileSystemWatcher1_Created(sender  As  Object,  e  As  FileSystemEventArgs)  Handles
FileSystemWatcher1.Created '在监控路径下添加文件时触发
            '这句代码的功能是延时，本人经过反复测试，发现复制单个文件到监控路径下，程序合并数据没有问题。
'如果同时复制多个文件到监控路径下，合并时则会出错。但是如果在合并时延时几秒钟就可以解决此问题，因此
'采用此方法
            For item As Int32 = 1 To 10000
                System.Windows.Forms.Application.DoEvents()
            Next item
            app.ScreenUpdating = False                          '关闭屏幕更新，从而提升代码执行效率
            Wbk = app.Workbooks.Open(PathStr & e.Name, False)   '打开被监控路径下的新文件
            newSht = Wbk.Sheets(Wbk.Sheets.Count)               '将最后一个工作表赋值给变量 newSht
            Item = Item + 1                                     '累加计数器
            合并(PathStr & FileName, acSht, newSht)             '调用过程“合并”
```

```
        app.CutCopyMode = False                          '取消剪切模式
        Wbk.Close(False)                                 '关闭已经合并完成的工作簿
        app.Cells.EntireColumn.AutoFit()                 '自动调整列宽
    If Item = 7 Then                                     '如果已经合并了 7 个工作簿
        '删除"e:\合并工作簿"所有文件
         Shell("cmd.exe /c del e:\合并工作簿\*.*    /q", AppWinStyle. MinimizedNoFocus)
            Me.Close()                                   '关闭工作簿(存放结果的工作簿)
           '在最后一个工作表名的最后追加"合并完成"
            acSht.Name = Format(DateAndTime.Today, "yyyyMMdd 已合并完成")
            app.ActiveWorkbook.Save()                    '保存工作簿
            app.ActiveWorkbook.Close()                   '关闭工作簿
       End If
        app.ScreenUpdating = True                        '恢复屏幕更新
    End Sub
```

以上事件过程的功能是当 FileSystemWatcher1 控件发现用户向“E：\合并工作簿”中复制文件时，使用 Workbooks.Open 方法打开该文件，然后调用“合并”过程复制数据，完成后关闭该工作簿。当计数器 Item 累加到 7 时自动删除所有文件，并且在“产量总表.xlsx”中标注合并完成，最后保存关闭“产量总表.xlsx”。

8．在解决方案资源管理器中双击“ThisAddIn.vb”，然后在代码窗口中录入以下代码：

```
    Private Sub Application_WorkbookOpen(Wb As Workbook) Handles
    Application.WorkbookOpen
      If Wb.Name = "产量总表.xlsx" Then '如果当前打开的工作簿是产量总表
        '如果最后一个工作表的名字不是今天的日期加"已合并完成"
         If app.Sheets(app.Sheets.Count).name <> Format(Today, "yyyyMMdd 已合并完成") Then
          '如果文件夹"E：\合并工作簿"存在
           If My.Computer.FileSystem.DirectoryExists("E：\合并工作簿") Then
              Dim f As New Form1
              f.Show() '那么执行窗体 Form1
            End If
        End If
      End If
    End Sub
```

Application_WorkbookOpen 事件过程在打开工作簿时触发。由于本插件只需要在“产量总表.xlsx”中调用，因此需要使用 If 函数判断当前打开的文件是否为“产量总表.xlsx”，如果文件名称正确，同时最后一个工作表的名称不包含“已合并完成”，而且“E：\合并工作簿”文件夹是存在的，那么就调用 Form1 窗体，从而执行报表合并。

在本例中，合并数据不需要用到窗体，监控文件夹中的文件变化也不需要窗体，但是 FileSystemWatcher 控件不能独立工作，必须放在窗体中，因此本例设计了一个空白窗体，但是在代码的运行过程中用户是看不到窗体的。

9．选择菜单“项目”→“生成自动合并报表”。

10．将上一章用到的“安装文件模板.iss”和“1.ico”、“WizModernImage.bmp”复制到“C:\自动合并报表\自动合并报表\bin\Debug”，打开“安装文件模板.iss”，然后将文中用到的 19 个插件名称替换成“自动合并报表”。

11．选择菜单“构建”→“编译”，从而生成 Setup.exe 文件。

本例案例文件请参考：..\ 第 19 章\自动合并报表\

19.4.2 测试

测试本例的插件需要三份文件，一是上一节设计的安装程序，二是供测试的样本文件，位于“..\ 第 19 章\自动合并报表\测试数据”文件夹中，三是一个名为“产量总表.xlsx”的、用于存放结果的报表。

测试步骤如下：

1．双击 Setup.exe 文件，然后一直单击“确定”按钮或者“下一步”按钮，直到安装完毕。

2．在 E 盘新建一个文件夹“合并工作簿”，并且通过快捷菜单使其共享，并赋予可读可写的权限。

3．在其他电脑中打开本机的共享文件夹，然后存放两个文件进去，分别是“A 车间.xlsx”(见图 19-15) 和“B 车间.xlsx”(见图 19-16)。

	A	B	C	D	E
1	姓名	工号	产品	数量	
2	罗惠芬	SW001	改锥	166	
3	刘向东	SW041	电工刀	145	
4	刘丽鹃	SW029	电工刀	181	
5	陈维民	SW036	扁嘴钳	190	
6	张芳	SW027	单头扳手	152	
7	古政东	SW015	电工钳	183	
8	张惠芳	SW016	雕刻刀	169	
9	胡源源	SW039	多用扳手	152	
10	周泽民	SW003	钢丝钳	172	
11	陈维军	SW021	钢丝刷	189	

20170314 | 20170315 | 20170316

图 19–15　A 车间的产量日报表

	A	B	C	D	E
1	姓名	工号	产品	数量	
2	古兴军	SW048	机用锯条	173	
3	张持年	SW030	尖嘴钳	178	
4	洪运飞	SW024	电工刀	181	
5	洪满春	SW044	滚筒刷	159	
6	朱由校	SW040	海绵刷	187	
7	陈星明	SW019	划线器	153	
8	刘秀珍	SW049	活扳手	191	
9	吴明秀	SW032	六角镙丝	184	
10	张淑华	SW033	镙钉	156	
11	刘光明	SW014	镙帽	134	

20170314 | 20170315 | 20170316

图 19–16　B 车间的产量日报表

4．双击打开“产量总表.xlsx”，此时插件会自动到共享文件夹中搜索报表，找到两份报表后会自动执行合并，合并后的效果如图 19-17 所示。其中 A 列表示数据的来源，B 列及以后的区域用于存放日报表的数据。

5．到其他电脑中打开共享的文件夹“合并工作簿”，然后将“C 车间.xlsx”和“D 车间.xlsx”复制进去，此时“产量总表.xlsx”会自动监测到文件夹中有新文件进来，因而自动打开文件执行合并，全程无须人工干预。

6．继续将其他 3 份报表也放入“合并工作簿”文件夹中，此时“产量总表.xlsx”会继续合并新的文件，当 7 份文件合并完成后，它会删除 7 份报表，同时关闭“产量总表.xlsx”。

7．重新打开“产量总表.xlsx”，其合并结果如图 19-18 所示。

补充说明

1．FileSystemWatcher 控件除了监控文件夹中新增的文件，还可以监控文件夹中删除文件和修改文件的操作，因此本例的合并插件还可以做到更强大，例如用户在中途删除日报表时程序自动删除已经合并的数据，用户中途修改报表时程序自动修改已经合并到总表中的数据，这部分内容交给读者去完成，但需要提示的是删除文件和修改文件时对应的事件分别是 FileSystemWatcher1_Deleted 和 FileSystemWatcher1_Changed。

2．在本案例中日报表的标题只有 1 行，如果读者实际工作中的报表标题行数不同，那么可以相应地修改代码；共享文件夹的路径也有可能因用户的习惯而有所不同，请相应地修改代码；如果只合并数值，不需要公式，那么应该在合并前将日报表的数据都转换成数值，然后再合并；读者在实际工作中可能要合并的报表不是 7 个，可以相应地修改代码。当然，也可以在窗体中加

几个文本框，用于设置这些参数，从而使插件更加灵活，能适用更多变的环境。

	A	B	C	D	E	F
1	来自工作簿	姓名	工号	产品	数量	
2	A车间.xlsx	罗惠芬	SW001	改锥	167	
3		刘向东	SW041	电工刀	178	
4		刘丽鹃	SW029	电工刀	153	
5		陈维民	SW036	扁嘴钳	184	
6		张芳	SW027	单头扳手	154	
7		古政东	SW015	电工钳	162	
8		张惠芳	SW016	雕刻刀	187	
9		胡源源	SW039	多用扳手	161	
10		周泽民	SW003	钢丝钳	166	
11		陈维军	SW021	钢丝刷	162	
12	B车间.xlsx	古兴军	SW048	机用锯条	173	
13		张持年	SW030	尖嘴钳	178	
14		洪运飞	SW024	电工刀	181	
15		洪满春	SW044	滚筒刷	159	
16		朱由校	SW040	海绵刷	187	
17		陈星明	SW019	划线器	153	
18		刘秀珍	SW049	活扳手	191	
19		吴明秀	SW032	六角镙丝	184	
20		张淑华	SW033	镙钉	156	

20170315已合并完成 | 20170316

图 19–17　自动合并 2 份报表的结果

	A	B	C	D	E	F
53	H车间.xlsx	朱宏光	SW035	镙钉	145	
54		曹大年	SW013	美工刀	152	
55		洪兴	SW006	十字扳	177	
56		古珍珍	SW034	双头扳手	168	
57		朱成新	SW042	圆盘刷	154	
58		刘文丽	SW050	园嘴钳	188	
59		诸贵	SW046	电工刀	154	
60		张则栋	SW020	弯管器	183	
61		朱明秀	SW009	圆嘴钳	148	
62	G车间.xlsx	黄秀丽	SW012	海绵刷	157	
63		朱宏光	SW035	镙钉	162	
64		曹大年	SW013	美工刀	187	
65		洪兴	SW006	十字扳	148	
66		古珍珍	SW034	双头扳手	148	
67		张则栋	SW020	弯管器	166	
68		朱明秀	SW009	圆嘴钳	147	
69		朱成新	SW042	圆盘刷	190	
70		刘文丽	SW050	园嘴钳	162	
71		诸贵	SW046	电工刀	197	

20170315已合并完成 | 20170316已合并完成

图 19–18　合并完 7 份报表的结果

3．本插件的价值在于早上打开“产量总表.xlsx”，不用担心 7 个车间的报表在什么时候发送过来，程序会全自动监测文件夹的变化，每复制进来一个文件就自动合并一次，直到 7 个文件合并完成时自动保存并关闭文件，同时删除已经没有用处的 7 份报表。

在随书案例文件中，笔者还为读者准备了一个当前插件的操作视频，便于读者理解插件的功能以及效果预览。插件操作视频存放在“..\第 19 章\自动合并报表\插件操作演示.mp4”。

19.5　设计任务窗格

在使用 VBA 开发插件时无法设计任务窗格，VSTO 弥补了此缺憾，可以开发任务窗格，并在窗体中添加任意窗体控件，从而实现与表格的交互功能。

本例介绍任务窗格的设计思路，同时也为读者提供一个任务窗格模板。读者在实际工作中需要设计其他的任务窗格相关的功能时，可以借助本例的模板稍加修改即可。

19.5.1　设计

借助任务窗格可以设计出各式各样的辅助工具，满足多样的需求。本例开发的插件重点解决日期录入的需求，单击窗格中的日历控件从而实现在活动单元格中录入日期，具体设计步骤如下：

1．打开 Visual Studio 2015 软件，新建一个“Excel 2013 和 2016 VSTO 外接程序”，将上方的.Net Framework 版本号由 4.5.2 修改为 4，且取名为“任务窗格模板”。

2．选择菜单“项目”→“添加模块”，然后在弹出的对话框中单击“添加”按钮。

3．在模块中的“Module Module1”下一行插入以下代码：

```
Public app As Excel.Application = Globals.ThisAddIn.Application
```

4．选择菜单“项目”→“添加用户控件”，然后在弹出的对话框中单击“添加”按钮。

5．进入工具箱，将 MonthCalendar 控件拖到 UserControl1 的区域内，然后调整 UserControl1 的大小，使其刚好容纳 MonthCalendar1，效果如图 19-19 所示。

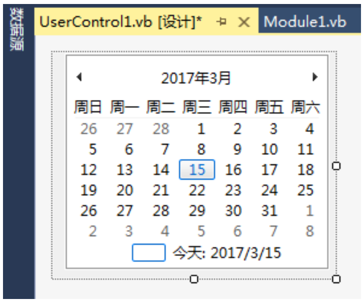


图 19-19　在用户控件内添加日历控件

6．在 UserControl1 旁边的空白区域单击右键，从快捷菜单中选择“查看代码”。

7．进入 UserControl1 的代码窗口后，在“Public Class UserControl1”与“End Class”之间插入以下代码：

```
'单击鼠标时触发此事件
Private Sub MonthCalendar1_MouseDown(sender As Object, e As
System.Windows.Forms.MouseEventArgs) Handles MonthCalendar1.MouseDown
    '将日历控件的起始值输出到活动单元格中（按住 Shift 键的同时选中多个日期时就涉及起始值
和终止值，如果只选择一个则这个值就是起始值）
    app.ActiveCell.Value = MonthCalendar1.SelectionStart.ToLongDateString
End Sub
```

8．在解决方案资源管理器中双击“ThisAddIn.vb”，然后在“Public Class ThisAddIn”下方录入以代码，用于声明 2 个公共变量。

```
Private 用户控件 As UserControl1
Public 任务窗格 As Microsoft.Office.Tools.CustomTaskPane
```

9．在空白的 ThisAddIn_Startup 事件中插入代码，使任务窗格可以随插件一同启动，完整代码如下：

```
Private Sub ThisAddIn_Startup() Handles Me.Startup
    用户控件 = New UserControl1        '将 UserControl1 赋值给变量用户控件
   '在 UserControl1 中创建一个 CustomTaskPane
    任务窗格 = Me.CustomTaskPanes.Add(用户控件, "日期输入器")
With 任务窗格 '引用 CustomTaskPane
     '显示在工作表左边
        .DockPosition = Microsoft.Office.Core.MsoCTPDockPosition.
msoCTPDockPositionLeft
        .Width = 240                   '显示宽度设置为 240
        .Visible = True                '让 CustomTaskPane 显示出来
    End With
    用户控件 = Nothing                 '释放变量的内存空间
End Sub
```

以上代码的功能是在加载插件的同时也加载任务窗格，任务窗格的位置是工作表的左方。msoCTPDockPositionLeft 修改为 msoCTPDockPositionRight 则表示任务窗格显示在右方。

通过以上步骤已经可以正常使用任务窗格了，不过为了方便控制任务窗格的显示状态，从而让操作时更人性化，还应该添加一个功能区菜单，单击菜单关闭任务窗格，再次单击菜单则显示任务窗格。

10．选择菜单“项目”→“添加组件”，选择“功能区（可视化设计）”，然后单击“添加”按钮。

11．在生成的功能区的右方或者下方单击右键，在弹出的快捷菜单中选择“删除”命令，从而删除内置的选项卡。

12．进入工具箱，分别将 Tab 控件、Group 控件和 Button 控件拖曳到功能区中， 将 Tab 控件和 Group 控件的 Label 属性值都修改为“录入日期”，并将按钮的 ControlSize 属性修改为“RibbonControlSizeLarge”，将按钮的 OfficeImageId 属性值修改为“StartAfterPrevious”，将按钮的 Label 属性修改为“已显示”。功能区按钮的效果如图 19-20 所示。



图 19-20　功能区按钮

13．双击“显示”按钮进入代码窗口，然后录入以下 Button1_Click 事件代码（过程外壳是自动产生的）:

```
Private Sub Button1_Click(sender As Object, e As RibbonControlEventArgs)
Handles Button1.Click
  '切换任务窗格的显示状态，单击显示再单击隐藏
Globals.ThisAddIn.任务窗格.Visible = Not Globals.ThisAddIn.任务窗格.Visible
    Button1.Label = IIf(Button1.Label = "已显示", "已隐藏", "已显示") '切换菜单文字
End Sub
```

14．保存代码。

本例案例文件请参考：..\ 第 19 章\发邮件模板\

19.5.2　测试

测试外接程序有两种方法，一是直接按【F5】键启动 Excel 2016，然后单击菜单测试代码，二是生成 DLL 插件，并按第 18 章的方法生成 exe 安装文件，然后安装好插件，再进行测试。

本节按第一种方法操作，步骤如下。

1．在 Visual Studio 2015 中按下【F5】键，从而打开 Excel 2016（如果在新建项目时选的是“Excel 2010 VSTO 外接程序”，那么将会打开 Excel 2010）。

2．在 Excel 的工作表左方出现一个名为“日期输入器”的任务窗格，单击日期，在活动单元格中就会产生该日期，操作效果如图 19-21 所示。

3．单击选项卡“录入日期”中的“已显示”按钮，任务窗格会马上隐藏起来，同时按钮的文字也会切换成“已隐藏”，效果如图 19-22 所示。

4．单击“已隐藏”按钮，任务窗格会马上显示出来。

通过以上测试，表明插件的功能满足需求。

若需要生成将插件打包成 exe 格式的安装程序，借用前面章节的模板和步骤说明，5 秒钟内即可打包完成，读者可以自行尝试。

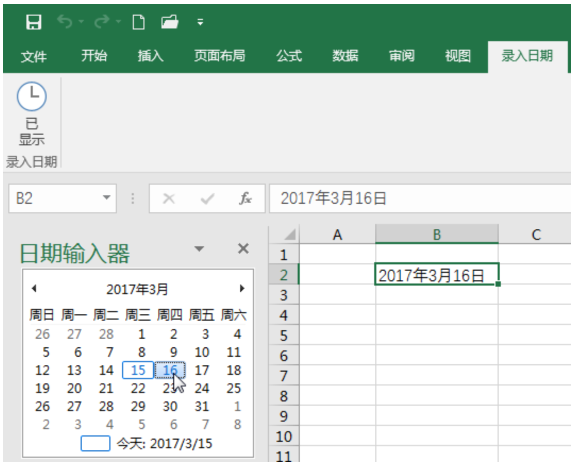


图 19–21　单击日历控件生成日期

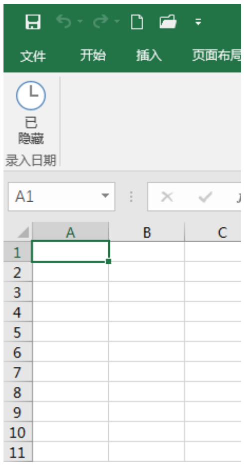


图 19–22　隐藏任务窗格

补充说明

1．本书采用 Visual Studio 2015 作为教学工具，但是书中的所有步骤和代码都通用于 Visual Studio 2010、Visual Studio 2015 和 Visual Studio 2017。

2．本书关于 VSTO 的教学思路是先用 VBA 编写插件，然后根据书中第 17 章的差异说明修改代码，使 VBA 代码能够用于 VB.net 中。不过由于两者的差异不算特别大，而且常见的需要修改的地方往往就 10 多个，在修改三五次代码之后就可以直接写书 VSTO 代码了，不再借助 VBA。常见的几个差异包含引用 Excel 对象时添加在前面添加“app.”，声明 Excel 对象类型的变量时在前面加“Excel.”，参数必须放在括号中，数组的下标为 0，在显示窗体时不能直接用“Userform1.Show”，而是先用 New 语句声明变量，然后再调用该变量去显示窗体等。

3．在 A 电脑上设计插件，再将文件复制到 B 电脑打开并生成 DLL 文件时，往往会提示：“无法在当前用户的 Windows 证书存储中找到代码签名证书。若要更正此问题，请禁用 ClickOnce 清单的签名或将证书安装到证书存储中。”

产生此提示后将无法生成插件，解决办法有两个，一是在解决方案资源管理器中双击“My Project”，然后在左方的选项卡中选择“签名”，单击“创建测试证书”按钮，接着录入用户名和密码，最后选择菜单“项目”→“生成您的插件名称”即可。方法二的不同之处在于如果以前已经创建过证书，那么在“签名”选项卡中单击“从储存区选择”按钮，然后选择一个已经存在的签名即可，不用重新创建证书，如图 19-23 所示。

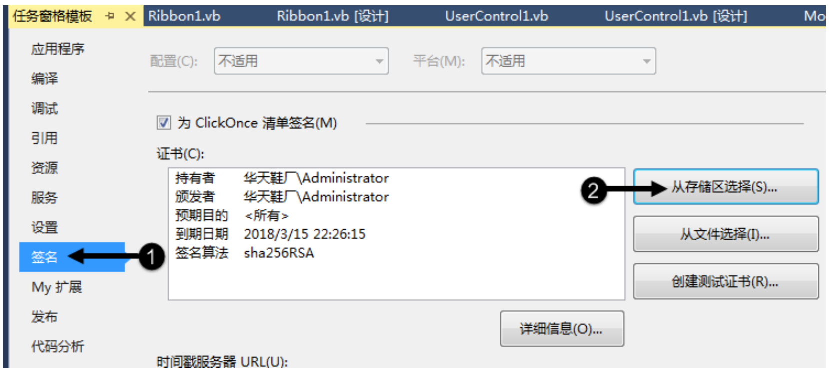


图 19–23　选择已经存在的签名

4．本书提供了读者交流群，群号为 47700194，在看书的过程中有不懂的地方可以加入 QQ 群与作者交流，或者与同学们交流，从而加快学习进度。

19.6　课后思考

1．有何办法能让用户在窗体文本框中录入字符时显示为三角形▲?

2．能否实现在双击 NotifyIcon 控件生成任务栏图标之后，自动隐藏该图标?

3．使用 Msgbox 函数提示用户“当前过程执行完毕”，该提示信息会一直在桌面上显示，直到用户主动关闭。有没有什么办法在通知用户 5 秒钟后，提示信息自动关闭?

4．代码“Globals.ThisAddIn.Application.CommandBars(1).Controls.Add”可以生成一个传统的工作表菜单，它和功能区按钮有何区别?

5．如何才能让任务窗格显示在工作表下方?

第 20 章　365 个 VBA 常见问题答疑

VBA 的知识面较广，应用范围也较广，为了拓展读者的知识面，在本书前 19 章的教学之外再向读者提供 365 个疑难解答。

为便于读者思考，书中仅提供问题，答案请到随书赠送的案例文件中查阅。

1．VBA 能开发 exe 格式的安装软件吗?

2．在编写 VBA 代码时，代码中所用的变量是否必须声明数据类型?

3．Dim 甲,乙,丙 as byte——这句 VBA 代码能声明三个 byte 类型的变量吗?

4．为什么有些电脑的“Visual Basic”和“录制宏”菜单呈灰色禁用状态?

5．为什么按【F8】键调试代码会失败，但是选择菜单“逐语句”却能进入调试状态?

6．打开 VBE 界面，看不到工程资源管理器，应如何调出工程资源管理器?

7．Dim FileSys As FileSystemObject——为什么在运行此声明语句时提示“用户定义类型未定义”?

8．如何实现 FSO 对象的后期绑定?

9．VBA 中如何获取当前用户的“发送到”文件夹地址?

10．VBA 中如何获取收藏夹路径?

11．为什么 VBE 中的菜单“视图”→“工具箱”是灰色的?

12．如何对 VBA 代码加密?

13．为什么有的属性在常数列表中看不到?

14．如何判断 Excel 2016 自带的 VBA 的版本号?

15．工作簿事件相关的代码是否在 Excel 2003、Excel 2007、Excel 2010、Excel 2013 和 Excel 2016 中通用?

16．是否只用低版本的方法、事件、属性就可以确保代码通用?

17．Excel 2016 相对 Excel 2010 在 VBA 方面的变化大吗?

18．VBA 可以实现不打开工作簿而取其中的值吗?

19．Excel 中的所有操作都可以录制宏吗?

20．单元格的颜色属性 Color 和 ColorIndex 是一样的吗?

21．相同公式，Excel 2016 会不会比 Excel 2003 计算更快?

22．如何使用 VBA 判断单元格是否含有超链接?

23．如何取得一个区域内的已用区域地址?

24．按地址传递（byref）和按值传递（byval）的区别是什么?

25．如何用 VBA 打开与陌生人的 QQ 聊天窗口?

26．如何判断当前操作系统是否为繁体中文?

27．VBA 可以禁止保存或者另存为吗?

28．可以通过 VBA 提取姓名的汉语拼音首字母吗?

29．能使自动替换功能只对一个区域生效吗?
30．如何获取 application.inputbox 引用的单元格所在的工作表名称?
31．Exit Sub 等于 End 吗?
32．利用 Public 声明的变量就是全局变量吗?
33．如何用 VBA 实现缩小功能区?
34．用 cells.count 计算单元格数量会出错，如何修改该代码呢?
35．Option Private Module 的作用是什么?
36．Function 过程的代码如何逐句调试?
37．在录制在单元格输入公式的宏时，产生的是 R1C1 样式的公式，如何转换成 A1 引用样式?
38．AfterSave 事件的参数 Success 是什么意思?
39．如何判断“Visual Basic for Application”是否拼写正确?
40．如何实现“工程不可查看”?
41．chr(10)和 chr(13)有什么区别?
42．VBA 可以打开资源管理器中的文件夹吗?
43．如何删除选区中的所有空格?
44．VBA 可以读写注册表中任意键值吗?
45．ontime 如何调用具有参数的过程?以下述代码为例。

```
Sub a(b)
  MsgBox b
End Sub
```

46．可以禁止用户插入新表吗?
47．如何中途阻止运行过程?
48．可以屏蔽内置的快捷键吗?
49．Range.text 与 Range.value 有何区别?
50．如何一键删除工作表中所有图形对象?
51．VBA 可以修改系统日期吗?
52．如何将工作簿另到 D 盘中，工作簿名由 A1 单元格的值决定?
53．如何禁用单元格的快捷菜单?
54．可以禁止双击修改单元格吗?
55．如何一次性清除工作表中所有公式结果为错误值的单元格?
56．Evaluate 方法可以将任意表达式进行转换计算吗?
57．可以为自定义函数添加每一个参数的说明吗?
58．如何生成包括“第 X 页，共 Y 页”的页眉?
59．如何判断当前工作表是否处于筛选状态?
60．可以用一句代码实现合并 A1:A10 单元格的值吗?
61．如何一键删除超链接?
62．可以用一句代码实现对二维数组变量赋值吗?
63．如何计算“D:\工作表”文件夹的大小?
64．不用循环，如何计算 1 到 100 之间的自然数之和?
65．可以用一句代码实现向下填充所有空白单元格吗?

66．什么是溢出错误?
67．什么是下标越界?
68．如何禁止在合并单元格和删除工作表时弹出对话框?
69．如何禁止用户中断程序的运行过程?
70．可以让比较单词时不区分大小写吗
71．可以让数组的默认下标为 1 吗?
72．可以用 VBA 打开控制面板吗?
73．如何获取“D:\123.xlsm”的创建时间?
74．动态变量和静态变量的区别何在?
75．如何防止程序因出错而弹出错误对话框?
76．批量删除或者插入行时，如何防止屏幕闪动?
77．运行窗体时可以只显示窗体，不显示 Excel 界面吗?
78．如何隐藏所有批注，包括红色箭头?
79．在对话框中录入字符串时，可以显示为星号吗?
80．如何取得 Excel 的安装路径?
81．如何打开 Windows 自带的计算器?
82．如何获取活动单元格的批注文本?
83．如何在状态栏显示今天星期几?
84．Worksheet_Change 事件中写入“Target = Evaluate(Target.Value)”用于转换表达式，为什么有时会出错?
85．什么是递归?
86．在 SheetChange 事件中，如何防止溢出堆栈空间?
87．如何取得已用区域最后一行的行号?
88．如何区分当前区域和已用区域?
89．什么是数组区域?
90．Range.End(3)中的 3 代表什么?
91．如何获取第一行最后一个非空单元格?
92．如何判断活动工作表是否空表?
93．为什么使用 Application.Inputbox 后返回值为是 Range 型?
94．自定义函数只能返回值，不能改变单元格的值吗?
95．Format 函数和工作表函数 TEXT 一样的功能吗?
96．VBA 能定制 Excel 启动画面吗?
97．如何实现在 11:00 时提示开会?
98．如何为“生成批注”过程指定一个快捷键?
99．可以让窗体中的文字框换行吗?
100．能否判断单元格的值和其显示的值是否一致?
101．如何判断活动单元格是否存在条件格式?
102．如何用一句代码删除 A 列的重复值?
103．可以使 Sub 过程不出现在宏对话框中吗?
104．如何计算活动工作表能打印多少页?

105．可以在每次打印前将已用区域更新为打印区域吗?
106．名称也分作用域吗?
107．为什么在代码前面有的空几格，有些不空？是必须的吗？有何规则?
108．能否将多句代码写在一行中?
109．将多句代码写在一行和写在多行，在执行过程中有任何区别吗?
110．一句较长的代码如何分多行书写?
111．有些代码中有“VBA.”，这是什么意思？是必须的吗?
112．某些代码前有一个撇号，它代表什么?
113．VBA 如何调用 DOS 命令?
114．用代码删除文件是否放入回收站?
115．如何在指定的行插入一个分页符?
116．常量既然在程序运行中不产生变化，为什么不直接使用值本身而要用常量呢?
117．为什么部分函数无法通过 WorksheetFunction 调用?
118．可以在活动单元格的批注中添加一行当前日期吗?
119．如何判断一个图片是否在 D1 单元格?
120．“信任对 VBA 工程对象模型的访问”复选框有何作用?
121．如何计算带有“[备注]”的表达式？例如“1200*8[8 台诺基亚手机]+3400*5[5 台 iPhone4 手机]”。
122．如何获取某个盘符的序列号？例如 C 盘。
123．如何获取硬盘的物理序列号?
124．VBA 中的 IIf 和 If...Then...Else 有何区别?
125．IIf 是否和工作表函数 If 用法一致?
126．当单元格快捷菜单被禁用时，如何恢复?
127．有什么方法、属性或者函数能检查是否存在某个名称的工作表?
128．如何将一个区域的公式转换成值?
129．批注框的大小可以调整为自动适应文字吗?
130．如何一次性将工作表中所有公式所在单元格标示为黄色?
131．如何启动发送邮件的窗口，且向指定用户名发送?
132．如何禁用窗体右上角的关闭按钮?
133．如何打开指定文件夹?
134．若【Alt+F11】快捷键被禁用了，如何修复?
135．可以把本工作簿的关闭时间写入注册表吗?
136．如何删除“D:\生产表”？
137．如何调用屏幕保护来锁定屏幕?
138．可以在执行代码时，暂停 0.5 秒再执行下一句代码吗?
139．可以一键删除所有“0”所在行吗?
140．如何用一句代码将 Sheet1 的所有数据复制到“总表”中?
141．如何将活动单元格最后一个字符设置为上标?
142．可以锁定所有公式所在单元格从而防止误修改吗?
143．如何取消工作表中所有单元格的隐藏属性?

144．如何隐藏所有零值?
145．若窗体中有两个文字框，能实现在文字框中录入数据后按【Enter】键执行命令，而不是通过单击按钮来执行命令吗?
146．窗体可以永远显示最上层吗?
147．如何删除 D 盘中所有 Excel 文件?
148．如何实现在状态栏显示实时更新的时间?
149．如何实现将选区数字的单位转换成以“万元”为单位?
150．如何一键添加已用区域的边框?
151．如何获取工作簿最后一次的保存时间?
152．如何取消所有加载宏?
153．可以列出工作表能用的所有字体名称吗?
154．如何计算开机到现在过去了多长时间?
155．如何实现关闭 Excel 程序但不弹出保存提示框?
156．可以通过 VBA 关闭计算机吗?
157．如何取消通过 Ontime 创建的任何计划?
158．可以通过“Esc”键关闭窗体吗?
159．如何让窗体显示在 Excel 窗体的正中间?
160．如何用两句代码选择已用区域的奇数行?
161．如何判断活动工作簿是否保存?
162．可以禁止使用 U 盘吗?
163．如何调用内置命令? 例如查找对话框。
164．可以获取本计算机的名称吗?
165．如何创建一个颜色选择框，并返回颜色值?
166．如何打开“定位条件”对话框?
167．A=10:B=10:C=10
Msgbox A=B=C
为什么以上代码得的结果不是 True?
168．若 VBE 界面中各窗口排列混乱或者某些窗口不见了，如何恢复默认设置?
169．能不能用 VBA 获知本工作簿的创建者?
170．Excel 创建新工作簿时总是默认包括 3 个工作表，可以修改为 7 个吗?
171．能否在上午 10 点时用语音提示“开会时间到”?
172．可以获取 system32 目录的路径吗?
173．如何创建一个按【Delete】键无法删除的文件夹?
174．通常工作表中所有公式都不显示计算结果，可以让它显示计算结果而不是公式本身吗?
175．可以让 Excel 2016 默认保存文件为 Excel 2003 格式吗?
176．可以取得 D 盘的剩余空间吗?
177．在 VBA 执行过程后，可以撤销该操作吗?
178．可以罗列出本工作簿定义的所有名称吗?
179．自定义函数时，如何为参数指定默认值?

180．如何定义参数个数不确定的自定义函数?
181．定义函数时，使用 ParamArray 需要注意什么?
182．在单元格中录入网址时会自动产生超链接，如何取消该功能?
183．Select 方法的参数是什么意思?
184．当本机中安装有多个打印机时，如何获取默认打印机名称?
185．VBA 能实现隐藏文件吗?
186．如何判断用户是否在兼容模式下使用 Excel?
187．如何判断今年是否润年?
188．如何禁用 Excel 的功能区?
189．在打开文件时弹出“安全禁告（宏已被禁用）”提示框，如何处理?
190．能让工作表中插入的图片产生单击或双击事件吗?
191．如何调用“选择和可见性”窗口?
192．可以让窗体中文字框中输入的字母总是小写状态吗?
193．可以修改工作表中的网格颜色吗?
194．可以终止 Excel 进程吗?
195．“录制宏”按钮为灰色禁用状态，其他按钮是正常的，怎么回事?
196．在录制宏时提示“不能记录”怎么回事?
197．可以暂时禁用鼠标操作和利用键盘在单元格录入任意字符吗?
198．可以让单元格的值按显示的值参与计算吗?
199．如何获取当前操作系统的版本?例如 Windows XP、Windows Vista、Windows 7。
200．用什么方法获取控件的属性值?
201．如何禁止使用快捷菜单中的“删除”命令?
202．如何让宏自动运行?
203．在 VBA 中“\”和“/”两个运算符有什么区别?
204．单元格不能使用填充功能，如何修复?
205．符号“+”在 VBA 中表示加法算法吗?
206．True 在参与运算时表示 1，False 在参与运算时表示 0 吗?
207．Range.Select 和 Range.Activate 有什么区别?
208．可以不显示最近打开的文件列表吗?
209．可以知道在本机中有哪些磁盘吗?
210．如何让工作表的单元格根据字符数量调整为最合适的行高?
211．可以通过变量的循环执行一系列 Sub 过程吗?例如宏 1、宏 2、宏 3。
212．如何判断是否添加了 scrrun.dll 文件的引用?
213．如何计算工作簿中的窗体、模块、类模块的总数量?
214．“编译错误：找不到工程或库”是什么意思?
215．可以在使用工作表打印时上下、左右居中吗?
216．Msgbox 的第二参数 64 代表什么意思?
217．Sheets 和 Worksheets 有什么区别?
218．为什么在使用自定义函数时，没有自动更新计算结果?
219．如何取消显示的分页符?

220．当宏的安装性设置为低时，可以做到打开工作簿而不启用宏吗？
221．如何判断变量的类型？
222．在 VBA 中“[]”括号在配合运算表达式使用时有什么用？
223．可以一次性插入 30 个工作表吗？
224．对普通变量赋值和对对象变量赋值有什么区别？
225．如何理解变量的作用域？
226．变量有生存周期吗？
227．如何让工作表区域全屏显示？
228．如何将区域转换成图片放入剪贴板，供程序调用？
229．ActiveWorkbook 与 ThisWorkbook 有什么区别？
230．如何判断工作表是否处于保护状态？
231．为什么使用“Cells.Find(What:="A").Activate”查找包括 A 的单元格时，有时成功有时不成功？
232．为什么“Range("A1048576")”运行有时出错，有时却可以？
233．如何获取两个区域的合集？
234．如何获取两个区域的交集？
235．Auto_Open 过程与 Workbook_Open 事件过程有什么区别？
236．为什么在工作簿保存后再打开代码就不见了？
237．录制宏有哪些限制？
238．可以取消双击受保护的工作表时弹出的警告对话框吗？
239．如何设置每 3 分钟自动保存？
240．循环语句中的 Step 有什么用？
241．Show 1 和 Show 0 有什么分别？
242．Debug.Print 是什么意思？
243．使用正则表达式需在添加什么引用？
244．VBA 中的正则表达式和其他软件中的正则表达式一样吗？
245．在正则表达式中，条件“A[BCD]E”支持哪些匹配？
246．在正则表达式中，匹配条件“\D”代表字母 D 吗？
247．在正则表达式中的“*”“?”两个元字符和 DOS 中的两个同名通配符功能一样吗？
248．VBA 中的 replace 和工作表函数 replace 功能一样吗？
249．VBA 代码也区分 32 位和 64 位吗？
250．可以用 VBA 可以开发商业插件吗？
251．在一列中取唯一值的方法很多，用什么方法通用性最好，且速度也快？
252．用 VBA 可以调用功能区中的任何功能按钮吗？
253．在利用 OfficeCustomUIEdito 开发功能区选项卡时，所有代码都区分大小写吗？
254．自定义功能区选项卡的作用对象是怎样的？
255．可以暂时关闭定制的功能区选项卡吗？
256．如何取得内置选项卡的按钮 ID？
257．可以用 VBA 修改自定义功能区的按钮图标吗？
258．可以用自己的照片作为功能区的按钮图标吗？

259．用 VB 能封装带有创建功能区菜单的 VBA 代码吗?
260．通过 OfficeCustom UI Editor 能定制哪些项目?
261．可以定制随选区不同而自动变化的功能区下拉菜单或者按钮吗?
262．可以利用 OfficeCustom UI Editor 在定制快速访问工具栏时保留选项卡吗?
263．能否删除 Office 文件菜单?
264．在内置的对话框启动器中可以显示图片，XML 语言定制的新启动器也可以实现吗?
265．在定制功能区选项卡时，对组、按钮或者下拉菜单的 ID 赋值有什么要求?
266．可以将工作表中插入的图片作为菜单图片吗?
267．如何通过工作表中的按钮执行录制宏?
268．在 Excel 自带的对话框启动器中可以显示图片，在自定义功能区的对话框启动器中也可以显示图片吗?
269．代码提速通常从哪些方面着手?
270．在封装代码时只能封装成 DLL 格式吗?
271．COM 加载项在执行过程中如果中断了，重新启动时会再也找不到其菜单项，如何再次调用菜单?
272．找不到“开发工具”选项卡，怎么办?
273．表单控件与 ActiveX 控件有何区别?
274．为什么以前可以使用 FileSearch 进行文件搜索，现在不可以了?
275．为什么采用 Trim 函数去除空格后，字符串中仍然有空格?
276．只想去除字符串首尾空格而忽略中间的，如何写代码?
277．“Is Nothing”代表什么?
278．如何不用循环判断一个一维数组 Arr 中是否包括字母“a”？
279．为什么两个看起来颜色不同的单元格，在用 Colorindex 取颜色值时结果却一样?
280．为什么 Excel 2003 可以通过“CommandBars(32).Controls.Add”在“分页预览”的快捷菜单中添加选项，而在 Excel 2010 或者 2016 中不可以?
281．加载宏用 xla 格式更好还是 xlam 格式更好?
282．加载宏如何安装?
283．Excel 2003 的快捷键是否可以用到 Excel 2016 中?
284．Excel 2003 的状态栏快捷菜单可以定制，Excel 2016 的也可以吗?
285．可以做到禁用宏则退出工作簿吗?
286．如何才能知道 Range 有哪些属性和方法?
287．通过 cells 和 Range 方式引用单元格有何区别?
288．为什么在调用硬盘中的 ICO 图标作为功能区按钮的图标时，偶尔会不成功?
289．为什么在 Excel 2016 中通过 VBA 代码插入图片后，文件发送其他人后，为什么收件人打开看不到图片?
290．工作簿中可以嵌入背景音乐吗?
291．工作中的控件为何删除不掉?
292．功能区相对老式菜单有什么优越性?
293．为什么同一个文件保存为 xls 格式和 xlsm 格式其大小相差很多?
294．用迅雷从网上下载 xlsx 或者 xlsm 格式的工作簿时，有时双击打开时不是 Excel，而是

WinRAR 这是怎么回事?
295．在定制功能区时，可以将控件、按钮直接放在选项卡中吗?
296．能添加新的选项卡，也能添加新的上下文选项卡吗?
297．VBA 中的 Date 和工作表函数 Date 有什么异同?
298．在 Excel 中选择多个不规则区域时禁止复制和剪切，VBA 可突破这个限制吗?
299．学会 VBA 后，以后 Office 升级是否需要再次学习 VBA?
300．在定制功能区选项卡时最好先写 VBA 代码还是先定义 XML 代码?
301．功能区可以缩小，能否像 Excel 2003 的菜单那样拖到工作表下方显示呢?
302．OfficeCustom UI Editor 软件没有查找与替换功能，写代码时需要批量替换怎么办?
303．定制功能区时，代码中“<”、“>”、“/”有什么含义?
304．动态菜单 DynamicMenu 能使菜单的样式、数量、名称等产生动态变化，创建动态的组吗?
305．在定义功能区的 XML 代码中创建注释时，用什么格式?
306．在定义功能区的 XML 代码是否和 VBA 代码一样，只能保存在 xlsm、xlam 格式的工作簿中?
307．如何确定回调函数的参数?
308．当定制的功能区中包括很多个组和多个按钮时，启动工作簿会不会很慢?
309．如何使用自定义图片作为功能区按钮的标图，图片的大小有何要求?
310．公式的计算速度快还是 VBA 更快?
311．VBA 代码要区分大小写吗？为什么书中或者网上的代码大多是首字母大写其余小写?
312．开发插件，必须使用菜单或者功能区按钮吗?
313．使用类模块最大的好处是什么?
314．在鼠标移过单元格时有对应的事件吗?
315．在工作表中插入图表时也有相应的事件吗?
316．是否在进行数据运算时尽量调用工作表函数?
317．Left$和 Left 有何区别?
318．如何使 Msgbox 对话框在提示用户的同时，让 VBA 代码继续执行?
319．能否实现在单击超链接时，即使所链接的工作表属于隐藏状态也可以正常跳转到目标工作表?
320．为什么“Sheets(3).Range("A2").Select”会执行不成功?
321．自定义函数的所有参数都是可选参数吗?
322．能用代码生成代码吗？即根据不同条件产生不同的代码，然后执行所产生的代码。
323．如何调用加载宏工作簿中的数据？例如 A1 的值。
324．VBA 可以实现屏幕截图并将该图片插入工作表吗?
325．如何记录某个工作簿的开启次数?
326．为什么将宏的安全性调低，保存文件时仍会出现“无法在未启用宏的工作簿中保存以下功能……”?
327．为什么将宏的安全性调低后，代码时仍然不执行，提示“不信任到 Visual Basic Project 的程序连结”?
328．为什么在“工程属性”对话框中录入查看工程属性的密码，且保存工作簿后仍然可以

看到 VBA 代码?
329．Auto_Open 过程与 Workbook_Open 事件有区别吗?
330．可以使工作簿只能在指定的电脑中开启吗?
331．可以用 VBA 代码生成一个新的功能区选项卡吗?
332．可以在 Excel 2010 的“文件”菜单中添加子菜单吗?
333．能用 VBA 定制 Mini toolbar 吗?
334．在录入定制功能区的代码时，如需在按钮的屏幕提示中使用换行符，用什么表示换行符?
335．如何用 VBA 打开 Word?
336．如何在关闭 Excel 程序时不提示保存工作簿?
337．在自定义函数时如何获取公式所在单元格的地址?
338．“Userform1.show 0”代码中的 0 是什么意思?
339．当运窗体时提示“显示有模式窗体时，不能显示无模式窗体”是什么意思?
340．工作簿“生产表.xlsm”的模块 1 中有一个过程“ABC”，如何在工作簿“财务报表.xlsm”中调用该过程?
341．能否将公历日期转换成农历日期吗?
342．能否禁止插入图表?
343．在 VBA 中等号是什么意思?
344．instr 和 InStrRev 函数有什么区别?
345．为什么“MsgBox True + 2”结果等于 1?
346．如何开启指定的网页?
347．为什么本机调用日历控件生成日期可以正常运行，工作簿复制到其他电脑后却不能运行?
348．为什么某些工作簿没有设置密码，却不能查看代码?
349．如何计算一个数组中的最大值?
350．在窗体中可以播放 Flash 动画吗?
351．列表框如何添加列标题?
352．用什么方法可以批量合并 A1:B1、A2:B2、A3:B3、A4:B4……
353．VBA 能调用 DOS 命令，是否也能格式化磁盘?
354．对 VBA 代码创建菜单时使用传统的菜单更好还是功能区按钮更好?
355．为什么以下代码中第二句会出错? 变量 i 的值在 100 到 200 之间，并没有超过 byte 的范围。

```
Dim i As Byte
For i = 200 To 100 Step -1
```

356．将代码封装成 DLL 后，有些什么优点?
357．在用 VB 封装代码时可以使用图片作为功能区按钮的图标吗?
358．For Next...Next 和 For Each...Next 两种循环方式有何区别?
359．录制宏对于学习 VBA 有什么帮助?
360．如何实现在单元格中输入自定义函数时也产生参数提示?
361．可以不打开工作簿而获取工作簿中某个工作表已用区域的值吗?

362．在利用 VB 6.0 企业版封装有窗体的 VBA 代码时，使用 VB 新建一个窗体更好还是直接将 Excel 中设计好的窗体导入到 VB 中更好?
363．VB 中有哪些函数或者语句与工作表函数同名但用法不同?
364．为什么 For Next 循环结束后，变量的值不等于循环体的最终结果（即 End 参数）?
365．在短期内 VBA 会被 VSTO 取代吗?